Universidad Carlos III de Madrid
Escuela Politécnica Superior

Departamento de Informática

Modelado de escenarios y posicionamiento basados en RFID

Proyecto Fin de Carrera
Ingeniería Técnica en Informática de Gestión

Autor: Borja Fernández Ruiz
Tutora: Jessica Rivero Espinosa
Septiembre 2010

Agradecimientos

En primer lugar y de forma muy especial a mis padres, Matías y Puri, sencillamente porque gracias a su educación, cariño y apoyo moral y económico, he llegado a ser quién soy. La educación que me han dado tiene como principios fundamentales el esfuerzo, la constancia en el trabajo, el respeto hacia los demás y la humildad, por lo que me siento tremendamente orgulloso de ellos. Creo que nunca podré recompensar todo lo que han hecho por mí, por lo que este 'éxito' se lo merecen tanto o más que yo.

A mi novia, Tere, por hacerme sentir más feliz cada día, por estar conmigo en los buenos y en los malos momentos y por darme todo su apoyo y confianza en cualquier aspecto y sobre todo durante estos últimos años de carrera y proyecto. No puedo explicar con palabras todo lo que siento por ella y la felicidad que ello supone. "Siempre juntos".

A mi familia, por su cariño y apoyo, por ver en mí un ingeniero técnico, motivarme a seguir adelante y hacerme disfrutar siempre a su lado.

A mis amigos/as, en especial a los más cercanos, por escucharme, apoyarme, animarme, compartir éxitos y grandes momentos además de hacerme valorar la amistad como si de un tesoro se tratase.

A mis amigos/as conocidos en la Universidad, compañeros de prácticas muchos de ellos, les agradezco su apoyo, colaboración y los grandes momentos que hemos pasado dentro y fuera de la Universidad. También significan para mí un tesoro de valor incalculable.

A mi tutora de proyecto, Jessica, por facilitarme la oportunidad de realizar este proyecto y por la brillantez con la que ha dirigido este proyecto gracias a su constante ayuda y atención.

A mis compañeros de trabajo, por su confianza, cariño y ayuda depositada, amén de todo lo que estoy aprendiendo junto a ellos.

A las personas que ya no están, en especial a mis abuelos y a Eduardo, porque no hay día que no me acuerde de ellos. Siempre estarán en mi memoria y en mi corazón.

A todos, muchísimas gracias por haberme acompañado durante estos años, no habría podido llegar hasta aquí sin todos vosotros.

Índice de contenidos

1. Introducción

La identificación por radiofrecuencia (RFID) es una tecnología de captura e identificación automática de información contenida en etiquetas electrónicas (tags) [1]. El objetivo principal de esta tecnología es transmitir la información de dichas etiquetas electrónicas mediante ondas de radio.

Las etiquetas RFID son unos dispositivos, generalmente pequeños, que en algunos casos son similares a una pegatina que se pueden adherir a distintos productos. Contienen antenas que permiten recibir y responder a las peticiones por radiofrecuencia de un lector RFID.

Esta tecnología ha tenido en los últimos años mucho auge debido a los grandes avances conseguidos, a la reducción de costes y precios en el mercado, así como a la comodidad de su funcionamiento para diversas aplicaciones.

Actualmente, una de las aplicaciones más importantes de RFID es la logística ya que el uso de esta tecnología permitiría, en las cadenas de suministro, localizar los productos y conocer información sobre ellos como por ejemplo dónde han estado almacenados y cuanto tiempo.

El sector textil también se beneficia del uso de esta tecnología. La inclusión de estas etiquetas en la ropa permite la optimización de recursos en los procesos de dispensación reduciendo el extravío y robo de prendas. Este uso resulta análogo a la inclusión de etiquetas en algunos productos de grandes almacenes, que facilitan el seguimiento de los mismos para evitar también robos y optimizar el proceso de reposición de productos.

Como se puede observar, aunque esta tecnología se encuentra en pleno proceso de evolución, son varios los usos ya implantados así como las futuras aplicaciones que se pueden beneficiar de las ventajas de esta tecnología.

En el siguiente apartado se detalla la utilidad del uso de esta tecnología y las posibilidades que brinda.

1.1. Contexto

Como se indicaba en el apartado introductorio, esta tecnología RFID está evolucionando poco a poco, gracias a los grandes beneficios que proporciona. Debido a esto, se prevé que el uso de esta tecnología tenga un impacto aún mayor que el actual sobre la actividad diaria y cotidiana de empresas, instituciones e incluso ciudadanos.

La tecnología RFID se basa en la comunicación por radiofrecuencia entre etiquetas electrónicas y lectores de las mismas. Para conseguir la información que tienen almacenada en su memoria las etiquetas se precisa un lector RFID, que realiza la función de emisor-receptor, de forma que cuando una etiqueta se encuentre en el área de cobertura o influencia del mismo, la etiqueta pueda trasmitirle (vía radiofrecuencia) dicha información . Ésta suele ser normalmente su identificador, que es un número de serie único (Auto ID) de 16 dígitos, y el valor de la potencia existente entre ambos elementos, es decir, el indicador de fuerza de señal de recepción. Este indicador, en inglés RSSI (abreviatura de Receive Signal Strength Indication) se mide en dBm, que se define como el nivel de potencia en decibelios en relación a un nivel de referencia de 1 mW [2]. Actualmente y gracias a su evolución, las capacidades de esta tecnología están mejorando y madurando en algunos aspectos como, por ejemplo, el rango de alcance (lectura). Inicialmente, era preciso que la distancia entre el lector y las etiquetas fuera corta y que hubiera visión entre ambos elementos. Actualmente, es posible encontrar soluciones que permitan lecturas de gran fiabilidad a varios metros, como se verá en este proyecto. Además de este aspecto, el rango de aplicación de esta tecnología se ha visto ampliado en los últimos años: los ejemplos citados en el apartado anterior, el telepeaje (recaudación con peaje electrónico) en autopistas, el seguimiento de equipaje en aerolíneas, el control de acceso en edificios, el seguimiento de historiales clínicos de pacientes hospitalarios, etc.

Como se puede observar y como veremos en el siguiente capítulo, son muchas las aplicaciones susceptibles de utilizar esta tecnología, que actualmente se encuentra en pleno proceso evolutivo.

Éste ha sido uno de los motivos que han motivado el uso de la tecnología RFID para este proyecto, en el que se propone un sistema de posicionamiento y localización. El objetivo

principal de este sistema es crear una red posicionando al usuario en un determinado escenario, con la posibilidad de crear enlaces entre los distintos posicionamientos, por lo que previamente se debe realizar un proceso de identificación o señalización del espacio (escenario), de tal forma que un perfil de usuario Administrador podrá crear puntos que identifiquen una posición en el escenario para que, posteriormente, el usuario de perfil Consultor pueda saber dónde está localizado dicho usuario.

A este sistema se le pretende dotar de robustez, escalabilidad, con una interfaz de usuario intuitiva y amigable, así como la capacidad de almacenamiento de información en base de datos con atributos geo-espaciales. Además, como se detallará en el siguiente capítulo, para los sistemas de localización y posicionamiento mediante RFID aún queda mucho por investigar.

El funcionamiento del sistema creado en este proyecto se basa en la comunicación e intercambio de información que se establece entre las etiquetas y el lector RFID y la conexión de éste último al PC, por medio del puerto de serie RS-232, con el objetivo de transferir los datos a la aplicación.

1.2. Plan de trabajo

Para la elaboración del Proyecto, éste se ha dividido en una serie de etapas, como una guía que facilitase y organizase todas las tareas que se han llevado a cabo. Estas etapas han sido, fundamentalmente, las siguientes:

1. **Estudio sobre la tecnología RFID. Introducción y conocimiento sobre cómo funciona y qué utilidades y posibilidades brinda.**
   Ante el desconocimiento de esta tecnología, relativamente moderna, sobre todo en el ámbito académico, era preciso comprender los conceptos fundamentales de la misma, así como sus limitaciones, posibilidades, ventajas y desventajas frente a otras alternativas.

2. **Componentes hardware de la tecnología RFID. Protocolos de comunicación. Cómo obtener información de dichos elementos.**

Una vez comprendida esta tecnología, era preciso conocer los protocolos de comunicación que utiliza y la forma de obtener la información del hardware disponible.

3. **Estudio de las tecnologías específicas utilizadas: JAVA y librería gráfica Swing. Base de datos Oracle Spatial.**
   Antes de comenzar con el diseño del sistema se estudian las tecnologías y herramientas que se van a utilizar para implementarlo.

4. **Análisis de casos de uso y requisitos.**
   Se procede a la creación de los casos de uso y al análisis de requisitos que deben llevarse a cabo para satisfacer las necesidades del sistema.

5. **Implementación de la base de datos e interfaces gráficos. Conexión entre ambos.**
   Se lleva a cabo la implementación de la aplicación. Se generan los scripts necesarios de bases de datos y se implementan las clases java que tienen como resultado los interfaces gráficos de la aplicación. Éstos deben estar conectados a la base de datos que almacene la información.

6. **Elaboración de la memoria.**
   La descripción del proyecto realizado se presenta en la memoria del mismo. Ésta se ha realizado, en su parte más amplia, al final de todo el proceso aunque resulta de vital importancia las anotaciones, comentarios y escritos realizados a lo largo de la elaboración del proyecto.

1.3. Objetivos

Los principales objetivos de este proyecto son, en primer lugar, realizar posicionamiento de usuarios en un determinado espacio en el que además se creará una red gracias a la unión de enlaces que permitirían guiar usuarios en un futuro. Con la implementación de este sistema se pretende otorgar una solución y además validez al uso de la tecnología RFID para esta problemática de posicionamiento.

Por tanto, en cuanto a la tecnología RFID:

- Se pretende proporcionar un protocolo de comunicación basado en RS-232 que permita a un PC conseguir información de etiquetas RFID por medio de un lector y básicamente, conocer con mayor profundidad lo que esta tecnología ofrece.

- Definir las ventajas y desventajas de esta tecnología frente a otras posibles soluciones de posicionamiento.

- Definir una correcta arquitectura adaptada al sistema permitiendo su escalabilidad e integración.

- Como se indicaba anteriormente, desarrollar un sistema que permita realizar posicionamiento de usuarios en un determinado escenario, utilizando esta tecnología por radiofrecuencia.

- Además de esto, el sistema permitirá crear una red, en base a dicha información de posicionamiento, que pueda servir como guía a los usuarios

En cuanto al desarrollo software, se pretende:

- Obtener los datos que el lector RFID puede extraer de la lectura de las etiquetas electrónicas.

- Implementar módulos software con interfaces gráficos intuitivos y amigables, que naturalmente permitan controlar y mostrar las operaciones de posicionamiento.

- Almacenar información geo-espacial en Oracle Spatial, ya que este hecho permite un interesante abanico de posibilidades.

1.4. Estructura del documento

El contenido de este proyecto está estructurado de la siguiente manera:

- **Capítulo 1. Introducción**. Descripción de la tecnología RFID y del marco de trabajo y el contexto que envuelven a este proyecto, etapas en las que se ha dividido y organización de la presente memoria.

- **Capítulo 2. Estado del Arte**. Se presenta el estado actual y evolutivo de la tecnología RFID, sus usos y aplicaciones que utilizan dicho tecnología. A su vez, se analizan otras alternativas para solucionar la problemática de posicionamiento que nos ocupa.

- **Capítulo 3. Análisis y Diseño**. Se detallan los requisitos del sistema y se estudian las funcionalidades que requiere el mismo. Se realiza una descripción del sistema, incluyendo los usos y los usuarios susceptibles de participar en el mismo. Se estudia la metodología utilizada y se realiza un diseño del hardware y software utilizado.

- **Capítulo 4. Implementación**. Se detalla en profundidad la solución realizada, es decir, la aplicación, desde el punto de vista técnico, describiendo la implementación de las clases software elaboradas así como de la Base de Datos.

- **Capítulo 5. Experimentación y resultados**. Describe el plan de Pruebas realizado y los resultados obtenidos después del mismo.

- **Capítulo 6. Conclusiones y líneas futuras**. Se describen las conclusiones extraídas del proyecto realizado, haciendo hincapié desde el punto de vista de la tecnología utilizada. Además, se exponen las líneas futuras de lo que podría ser algunas mejoras de la solución realizada en este caso.

- **Anexos**. Por último, se detallan los **acrónimos** utilizados, la **bibliografía** utilizada para la elaboración del proyecto así como algunos anexos complementarios como el **manual de usuario**, la **planificación** del proyecto y el **presupuesto** del mismo.

2. Estado del Arte

En este capítulo se realiza una breve introducción sobre la materia en la que se centra principalmente el presente proyecto, es decir, la importancia hoy día de los sistemas de localización, o posicionamiento, y su utilización. A su vez, se detalla de forma exhaustiva la tecnología que se utiliza en este proyecto (RFID), así como el análisis de algunas de las alternativas que también se podrían utilizar.

Por último, se analizan y detallan las tecnologías y herramientas utilizadas para implementar el sistema, en concreto, la base de datos Oracle, con el paquete Spatial para representar objetos geométricos, y el lenguaje de programación Java, con el software NetBeans y el lenguaje UML para modelado de sistemas.

2.1. Sistemas de localización o posicionamiento

En los últimos años se ha producido un crecimiento importante en la demanda de servicios de localización, surgiendo distintas tecnologías que demuestran una gran aceptación de estos sistemas así como un futuro optimista. Básicamente, el objetivo de estos sistemas es conocer la posición de una persona, objeto, etc. dentro de un escenario concreto. Normalmente, la mayoría de los sistemas utilizan un proceso de triangulación [3] para conocer el punto exacto, que se detallará más adelante. Para medir la distancia entre dos puntos de forma inalámbrica hay dos formas posibles: una de ellas es midiendo el retardo de propagación de una señal que viaja a una velocidad de propagación conocida, y la segunda es midiendo la atenuación de una señal por el efecto de la distancia.

Para el primer caso, el GPS es el ejemplo más representativo. Para el segundo, que será el utilizado en este proyecto, se precisa la identificación por radiofrecuencia. En este caso, la atenuación se mide a través de la señal RSSI (Received Signal Strength Indication), es decir, la intensidad recibida por parte de las etiquetas RFID. Este concepto aparecerá en repetidas ocasiones a lo largo del presente documento.

2.1.1 Alternativas. Otras tecnologías de Posicionamiento

Como ya se avanzó en el apartado anterior, se ha optado por utilizar el sistema de identificación por radiofrecuencia. No obstante, hay variedad de alternativas que solucionan la problemática que suponen los sistemas de posicionamiento [4].

1.-GPS

El **GPS** (**Global Positioning System**: **sistema de posicionamiento global**) o **NAVSTAR-GPS** es un sistema global de navegación por satélite que permite determinar, en todo el mundo, la posición de un objeto, una persona, un vehículo, etc., con una precisión hasta de centímetros (si se utiliza GPS diferencial [5]), aunque lo habitual son unos pocos metros de precisión. Actualmente está al cargo de este sistema el Departamento de Defensa de los Estados Unidos [6]. El GPS funciona mediante una red de satélites que cubren toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se está utilizando para ello, localiza automáticamente, como mínimo, tres satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. En base a estas señales, el receptor sincroniza el reloj del GPS y averigua el ángulo en el que se encuentra el punto (posición) respecto a los satélites, así como la distancia a los mismos. Este proceso se conoce como triangulación. Conocidas las distancias y los ángulos del punto respecto a los satélites, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta, o coordenadas reales, del punto de medición.

2.- Sistemas biométricos

La **biometría** [7] es el estudio de métodos automáticos para el reconocimiento único de humanos basados en características físicas y/o de comportamiento, por lo que normalmente el propósito de la utilización de esta tecnología es el de autentificar la identidad de un individuo.

Las huellas dactilares, las retinas, el iris, los patrones faciales, de venas de la mano o la geometría de la palma de la mano, representan ejemplos de características físicas

(estáticas), mientras que la firma o el paso al andar ejemplifican ejemplos de características del comportamiento (dinámicas). La voz se considera una mezcla de características físicas y del comportamiento, pero todos los rasgos biométricos comparten aspectos físicos y del comportamiento.

3.- Código de barras

El **código de barras** [8] es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información. Actualmente, es la tecnología más utilizada por los comercios para identificar los productos.
Este tipo de identificación se realiza codificando datos en una imagen formada por combinaciones de barras y espacios. Las imágenes son leídas por equipos especiales de lectura óptica a través de los cuales se puede extraer información del mismo..

La correspondencia o mapeo entre la información y el código que la representa se denomina *simbología* [9].

4.- Bluetooth

Bluetooth es una tecnología (protocolo) de comunicación inalámbrica destinada a eliminar los cables de dispositivos fijos y móviles, facilitando la comunicación entre ellos, y manteniendo altos niveles de seguridad. Sus principales características son su robustez y su bajo consumo y coste [10] [11].

En cuanto a localización se refiere, esta tecnología se utiliza, entre otras cosas, para formar redes de localización. Cuando se usa para tal fin, el sistema rsultante se denomina Red de Localización Bluetooth (Bluetooth Location Network, BLN) que se basa, fundamentalmente, en la capacidad de los módems Bluetooth de “descubrir” la presencia de módems afines a su entorno, de forma que pueda conocer su localización [12].

5.-Wi-Fi

Wi-Fi es un conjunto de estándares para redes inalámbricas basado en las especificaciones de los estándares IEE 802.11 [13] [14].

Esta tecnología permite conectar distintos dispositivos entre sí (un ordenador, una PDA, etc.) sin necesidad de cables. Actualmente, la implantación de esta tecnología para el acceso a Internet se encuentra en auge, pues no es raro encontrar este tipo de conexión en hoteles, aeropuertos, cafeterías, autobuses [15], etc., de manera que es posible realizar una conexión en cualquiera de estos sitios a través del ordenador portátil, de la PDA o de cualquier otro dispositivo móvil.

En cuanto a localización se refiere, la tecnología Wi-Fi tiene diversas aplicaciones, aunque la más relevante es la conocida como Wi-Fi mapping, realizada por la empresa Ekahau [16], y utilizada, por ejemplo, en el Museo de Arte Contemporáneo de A Coruña donde, de forma similar al objetivo de este proyecto, se modela el escenario en función de la señal recibida en cada baldosa de cada sala, es decir, se mide la coordenada y la señal Wi-Fi en cada una de ellas. De esta forma un vigilante con una PDA en el bolsillo conectada a la red Wi-Fi puede conocer su ubicación dentro del museo.

2.1.2 Ventajas de la identificación por radiofrecuencia

A continuación se describen las ventajas más significativas de la tecnología RFID que han hecho, entre otros motivos, que sea la tecnología elegida para afrontar este proyecto. Algunas de las ventajas son las siguientes:

- Seguridad y unicidad. Las tarjetas RFID no pueden duplicarse fácilmente. Además, cada tarjeta tiene un código unívoco que las diferencia de las demás lo que evita duplicidades.
- Comodidad. Esta tecnología es muy práctica en el sentido de que no es preciso que los elementos estén completamente visibles, de forma que la interacción entre el lector y las tarjetas sea cómoda y eficaz. Además, el tamaño de los lectores y sobre todo de las tarjetas no es excesivo.
- Eficacia de los lectores y tarjetas RFID. Siguiendo el hilo del punto anterior, el hecho de que no sea necesario visualizar completamente las tarjetas hace que

éstas puedan ocultarse si es preciso. Además, puede instalarse a la intemperie puesto que soportan las temperaturas ambientales.

- Amplitud de funcionalidades y aplicaciones. El área de aplicación de esta tecnología es muy amplio. Además, su versatilidad, permite utilizarla para diferentes funcionalidades.
- Reutilizable. El tipo de etiquetas RFID utilizadas en este proyecto pueden ser leídas por diferentes lectores. Esto hace posible el que sean usadas como componente hardware de otros sistemas con funcionalidad similar. Además, la información que proporcionan (su identificador) es válida para cualquier otro contexto.
- Precio. Es una tecnología en auge cuyo precio empieza a resultar interesante para algunas aplicaciones.

2.2. Tecnología RFID

A continuación se pasan a explicar algunos aspectos relevantes de la tecnología de localización que se va a usar en este proyecto, haciendo un breve repaso de la historia de la misma así como de las características técnicas de la misma.

2.2.1 Historia

Al contrario de lo que pueda parecer en la actualidad por su creciente rango de aplicación, la tecnología RFID no es reciente, de hecho ha sido en estos últimos años, gracias, entre otros, a la reducción de costes, cuando se ha popularizado.

Diversas fuentes señalan diferentes fechas de origen de esta tecnología, aunque sí parece un hecho que a finales de la década de los 60 surgieron las primeras aplicaciones tecnológicas con RFID (se dice que en los años 40 se utilizó un sistema parecido a RFID en la Segunda Guerra Mundial).

Fue Mario Cardillo el que en Estados Unidos, en el año 1969, patentó la primera aplicación tecnológica con RFID.

Es a partir de la década de los 80 cuando la tecnología RFID llega a Europa como medio de identificación del ganado en el sector privado, equipajes e incluso de ciertos vehículos. Estas etiquetas originales eran complejos sistemas de bobinas de metal, antenas y cristales. Se alimentaban por campos magnéticos generados por el lector RFID [17].

A partir de la década de los 90 la tecnología RFID comienza a adquirir más relevancia gracias a la reducción de costes por parte de varias compañías [18]. IBM, por ejemplo, consigue integrar todo el circuito en un solo chip.

En estos últimos años, las etiquetas RFID y los lectores se han perfeccionado aún más, haciendo que la tecnología RFID se utilice para diversas funciones y aplicaciones tecnológicas, tal y como se verá más adelante.

2.2.2 Componentes RFID

Como se ha avanzado en el capítulo 1, la tecnología RFID se basa en la comunicación que se establece entre un transpondedor (etiqueta RFID) y un lector de RFID.
El lector intenta comunicarse, utilizando una cierta frecuencia, con la etiqueta RFID de tal manera que ésta, cuando se encuentra en el área de influencia del lector, le contesta con la información almacenada en su memoria que normalmente es un número que la identifica. En el caso de los componentes utilizados para este proyecto, las etiquetas RFID le envían al lector un número de identificación de 16 dígitos y el valor de la potencia existente entre ambos elementos (RSSI). El lector recoge la información y la envía a un subsistema de procesamiento de datos.

Un sistema RFID se compone de los siguientes componentes básicos [1]:

1. El tag, transpondedor o etiqueta RFID que consiste en un pequeño circuito con una antena. Es el encargado de enviar datos al lector de RFID. Algunas veces incluye una batería. Los diferentes modelos de tags tienen características en común pero también tiene características que las diferencian entre ellas.

- Lectura del tag: cualquier tag debe poder comunicar la información mediante radiofrecuencia.
- Kill/Disable: algunos tag permiten recibir una orden del lector para dejar de funcionar.
- Write Once (una sola escritura): como ya se ha dicho, muchos tags contienen su identificador internamente pero hay otros modelos que permiten al usuario escribir este valor una y solo una vez.
- Write Many (varias escrituras): algunos tags permiten escribir el campo de identificador tantas veces como se desee.
- Anticolisión: cuando hay varios tags cercanos a un lector, estos tags tienen la capacidad de comunicarse entre ellos para no entorpecer las lecturas con el lector, de tal forma que cada tag espera su turno.

2. Lector RFID o transceptor, que puede leer o escribir datos hacia los tags. Tiene los siguientes componentes: transmisor, receptor, antena, microprocesador, memoria, canales de entrada y salida, fuente de alimentación, controlador e interfaz de comunicación. El lector envía señales periódicamente hasta que capta una etiqueta, situada en su rango de alcance. Ésta le envía la información que tiene almacenada y el lector la recoge para enviarla a su vez al subsistema de procesamiento de datos.
3. Subsistema de procesamiento de datos: comúnmente se utiliza un PC, en el cual se tiene una base de datos para almacenar los datos y un software que utilice estos datos para diversas funciones.

Cabe destacar la importancia que tiene el modo de alimentación de los tags ya que éstos, en función de ello, pueden dividirse en tags pasivos, activos y semi-pasivos (o semi-activos):

- Tags pasivos: no cuentan con fuente de alimentación, por lo que la antena obtiene la energía de la transmisión del lector y será esta energía la que permita al tag enviar una respuesta. Dada la ausencia de fuente de alimentación estos dispositivos pueden ser muy pequeños y económicos pero tienen en contra que normalmente su rango de lectura es muy limitado, normalmente de unos 6 metros como máximo.

- Tags activos: al contrario que los anteriores, los tags activos cuenta con una fuente de alimentación incorporada como, por ejemplo, una batería. Además, tiene un tipo de electrónica más sofisticada que permite almacenar mayor cantidad de información. Otra característica importante es que su rango de lectura aumenta respecto a los tags pasivos. En este caso, dicho rango se encuentra entre 20 y 100 metros. En contraposición, estos tags son más costosos y de mayor tamaño al incorporar fuente de alimentación.
- Tags semi-pasivos: son muy similares a los tags pasivos salvo que incorporan una batería. Es esta batería la que permite al circuito integrado estar alimentado constantemente. Además, otorga la posibilidad de ser leídas a mayor distancia pues no necesitan la señal del lector para adquirir energía (al contrario que los tags pasivos).

2.2.3 Frecuencias

Las frecuencias RFID se dividen en los siguientes rangos [19]:

1. Baja frecuencia (9-135 KHz). Este rango de frecuencia tiene la desventaja de que el rango de lectura es muy pequeño, de sólo unos cuantos centímetros.
2. Alta frecuencia (13.56 MHz). Cubre distancia desde 1cm a metro y medio. Normalmente esta frecuencia es la que utilizan los tags pasivos.
3. Frecuencia ultra alta (Ultra High Frequency, 0.3-1.2 GHz). Este rango de frecuencia posibilita mayores alcances (de hasta 4 metros) y mayor velocidad de lectura. Esta velocidad permite, por ejemplo, cobro de peajes (también utilizado con frecuencia microondas).
4. Frecuencia microondas (2.45-5.8 GHz). Este rango está incluido en la banda de frecuencia UHF descrita en el punto anterior. Las etiquetas que soportan esta frecuencia tienen más potencia y por ende son más costosas. Su gran ventaja es el rango de lectura, que supera los 6 metros.

2.2.4 Estandarización

Como en otras áreas tecnológicas, la tecnología RFID debe cumplir con unos estándares previamente creados que “regulan” el uso de esta tecnología. Para este caso,

las principales organizaciones creadoras de estos estándares son ISO y EPCglobal de EPC (Electronic Product Code).

- ISO: ha desarrollado estándares de RFID para la identificación automática y la gestión de objetos. Existen varios estándares relacionados, como ISO 10536 [20], ISO 14443 [21] [22] e ISO 15693 [23], pero la serie de estándares estrictamente relacionada con los RFID y las frecuencias empleadas en dichos sistemas es la serie 18000 [24].
- EPC: EPC global ha desarrollado una amplia gama de estándares. Éstos están enfocados a la cadena de suministro, y definen la metodología para la interfaz aérea así como el formato de los datos almacenados en una tarjeta RFID [25]. EPC es un esquema de identificación de objetos físicos de manera universal por medio de etiquetas RFID. Esté código de EPC puede identificar al fabricante, producto, versión y número de serie, con dígitos extra para identificar objetos únicos. Además, EPC global ha trabajado con un estándar internacional para el uso de RFID y EPC, llamado Gen 2 (aprobado en 2004) [26] así como en un sistema (más antiguo) llamado ONS (Object Naming Service [27] [28], similar al DNS (Domain Name Service) utilizado en internet para direccionar equipos resolviendo nombres de dominios. ONS actúa como un directorio para organizaciones que desean buscar números de productos en Internet.

2.2.5 Conectividad

Cuando se utiliza un sistema con tecnología RFID se debe analizar la forma de conectividad de red del lector RFID. Normalmente, los lectores RFID utilizan conexiones seriales en RS-232, RS-485, y conexiones por puerto USB aunque actualmente se está potenciando la posibilidad de conectar los lectores vía Ethernet o Wireless.

La conexión por puerto RS-232 tiene como principal limitación la baja velocidad de comunicación, que va desde 9600 bps hasta 115.2 Kbps). No obstante, dependiendo de la aplicación para lo que se utilice el sistema RFID esta característica puede no ser de vital importancia. La comunicación es punto a punto, y no cuenta con control de errores.

Una mejora respecto a esta conexión es RS-485 que alcanza mayores velocidades y permite conectar más dispositivos al mismo cable.

Una tercera variante es el uso del puerto USB. Muchos proveedores de lector RFID permiten comunicar sus equipos por USB ya que el puerto serial está quedando obsoleto y está desapareciendo de los equipos informáticos.

Para este proyecto se ha utilizado un lector RFID que sólo permite la conexión por puerto RS-232. No obstante, con el objetivo de modernizar la arquitectura del sistema y poder usarlo en cualquier equipo, se ha utilizado también un convertidor de puerto USB, que hace la función de pasarela entre ambos puertos, creando un puerto virtual, como se detallará en el siguiente capítulo. Sin embargo, este sistema mantiene las características del puerto RS-232, es decir, no permite un gran alcance de comunicación y la velocidad de la misma se ha establecido en 115200 bps para que pudiese funcionar.

Como se decía anteriormente, los lectores RFID también se pueden conectar vía Ethernet o Wireless. Ethernet es una buena opción porque ofrece una velocidad adecuada y porque el protocolo TCP/IP asegura la integridad de los datos, además de que es la infraestructura común para las redes por lo que no supone un esfuerzo extra de integración en su arquitectura para una organización que desee utilizar lectores RFID con esta conexión. La conexión Wireless se utiliza en la actualidad en los lectores RFID móviles. Su coste es menor pues no se necesitan cables.

2.2.6 Rango de aplicación. Casos prácticos de uso

Una vez explicados y descritos los aspectos más teóricos de esta tecnología, a continuación se describen los principales sectores en los que se está utilizando. Esta información se completa con algunos casos prácticos, con lo que se puede comprobar las ventajas y beneficios del uso de RFID según el ámbito de aplicación.

En el **sector de logística y transporte** la utilización de esta tecnología permite tener localizado cualquier producto dentro de la cadena de suministro. Hay variedad de aplicaciones que han demostrado ser útiles como por ejemplo la gestión de flotas en las compañías de transporte, control en el transporte de productos o animales

(identificación de palés por ejemplo), mejora de seguridad en carretera o identificación de productos con el fin de evitar robos y falsificaciones al ser transportados.

En este sentido, la **cadena de suministro y el inventariado** es otro posible ámbito de uso de esta tecnología. El objetivo de dichas cadenas de suministro es canalizar la mercancía desde el fabricante hasta el usuario final. Para ello, resulta imprescindible controlar la mercancía que se está moviendo y tener un inventario de ella, con el fin de mejorar la eficiencia y evitar problemas como ausencia de productos, inexistencia real de esos productos, robos, etc.

Por tanto, se podrían utilizar tags por cada artículo con el objetivo de inventariarlo, tener un control sobre el mismo e incluso aportar mayor información sobre el artículo. En el caso de los vendedores finales de estos productos, como por ejemplo supermercados o grandes superficies, la utilización de RFID puede resultar interesante para controlar el número de existencia e información como la fecha de caducidad de cada artículo en escaso tiempo pues un lector puede recoger información de varios tags (artículos) en escasos segundos. De esta forma se reducen tiempos y costes a la hora de recoger información de los productos, se identifica cada artículo, se reduce la intervención humana y se tiene un mayor control sobre los artículos, aumentando por tanto la seguridad.

En el **sector público y en las Administraciones Públicas**, a la tecnología RFID aún le queda mucho camino que recorrer. Se considera que puede tener cabida en multitud de aplicaciones como, por ejemplo, en la lectura de placas de locales comerciales. Desde las Administraciones se guarda información sobre los locales comerciales que operan en la zona por lo que la utilización de tags podría ser interesante para guardar información sobre cada local y las actualizaciones que sobre él acontezcan.
Otra posible aplicación, en el caso de las Administraciones Públicas, ahora que los ciudadanos cada vez tienen más medios de acceso y comunicación con la Corporación Local, podría ser la utilización de tags individuales para guardar información sobre los trámites realizados por cada ciudadano.

En el **sector del comercio** también se ha utilizado esta tecnología. En Barcelona se presentó a finales del año 2008 un carro de compra "inteligente", equipado con un lector

de radiofrecuencia y una pantalla de plasma con el objetivo de conocer, entre otros, el precio de cada producto introducido en el carro, el importe global del mismo e información de nuevas promociones [29]

En la **industria sanitaria** el uso de la tecnología RFID también podría ser relevante. Por ejemplo, una pulsera identificativa RFID permitiría identificar a cada paciente e incluso tener localizado a cada uno de ellos en una determinada zona de un hospital [30].
Otra posible aplicación en relación a la industria sanitaria es identificar productos o prescripciones médicas de cada paciente.

Otro ámbito en el que se utiliza la tecnología RFID es en el **control de acceso**. Sustituyendo a las tarjetas de banda magnética por tarjetas de identificación RFID se puede controlar el acceso a edificios, así como realizar una gestión y control horario (fichaje) de empleados, etc.

Medios de transporte como los autobuses. Se han realizado pruebas con tarjetas RFID en el servicio de transporte público de autobús de algunos países como Brasil. Cada marquesina dispone de una antena que se comunica con las tarjetas RFID que llevan cada uno de los autobuses. En las marquesinas se disponen pantallas informativas que ofrecen al usuario información de los autobuses que van a llegar. Además, las tarjetas RFID ubicadas en cada autobús sirven también para validar el acceso al mismo de los usuarios.

Implantaciones en animales y humanos. Existen soluciones que dan cabida a la identificación de ganado [31]. El sistema se basa en identificación y rastreo de ganado. Captura datos sobre la vida y el movimiento del ganado a través de la cadena de suministro. Con ello se pretende mejorar la eficacia y calidad de los productos.

Similar a este sistema es la implantación de etiquetas RFID pasivas en animales [32]. Estas etiquetas se introducen bajo la piel del animal en forma de capsulas de cristal. Estas etiquetas poseen un identificador que identifica unívocamente al animal de forma que con un lector (en un veterinario o en una sociedad protectora de animales, por ejemplo, y gracias a una base de datos) puedan obtener información del animal y datos de su propietario, entre otros.

Por último, este tipo de implantaciones ya se han llegado a utilizar en humanos. El más interesante es el de la Policía de Ciudad de Méjico, que ha implantado un chip a algunos oficiales de policía que permiten poder seguirlos en caso de secuestro. Hay algunos otros casos en los que se ha utilizado esta tecnología RFID, básicamente para identificación.

2.3. Oracle Spatial

En el siguiente apartado se explican, de manera general, las características del paquete Spatial de las bases de datos Oracle que resulta imprescindible para poder almacenar datos espaciales. Además, se incluirá un estudio más detallado de los distintos tipos de geometría que proporciona (así como de la forma en la que se crean) y de las funciones más relevantes.

2.3.1 ¿Qué es Oracle Spatial?

Oracle Spatial, comúnmente conocido como Spatial, es una opción o paquete para las bases de datos Oracle que permite el procesamiento espacial dentro de cada una de ellas. Este componente incluye funciones y procedimientos que dotan de capacidad espacial avanzada a una base de datos, respaldando así aplicaciones geo-espaciales, servicios basados en localización y sistemas de información espacial.

Oracle Spatial supone una evolución tecnológica ya que mejora las características de otros sistemas. Es el caso de algunas soluciones de análisis y tratamiento de datos espaciales de bases de datos, como GIS, que constan de sistemas autónomos de información, haciendo que muchos sistema utilicen un tipo de arquitectura dual, con un almacenamiento de datos dedicado a los objetos espaciales y a algunos de sus atributos. La creación de estándares abiertos de datos espaciales ha mejorado la situación, surgiendo Oracle Spatial, que cumple los estándares abiertos de datos espaciales más importantes: OpenGIS [33] y SQL/MM Part 3 [34]. Por tanto, Spatial, siguiendo estos estándares, define un esquema estándar SQL que soporta el almacenamiento, recuperación, consulta y actualización de los datos espaciales dentro de la misma base de datos utilizada para almacenar cualquier otro tipo de datos, de forma que los datos espaciales se puedan relacionar con cualquier otro dato almacenado en esa misma base de datos. Estas características de Oracle Spatial se pueden resumir en las siguientes [35]:

- Un esquema (MDSYS) que permite el almacenamiento, sintaxis y semántica de los tipos de datos espaciales (geométricos).
- Mecanismo de indexación espacial.
- Contiene operadores, funciones y procedimientos para realizar consultas espaciales, así como otras operaciones de interés.
- Facilita un modelo de datos (topología) para trabajar con datos sobre nodos, aristas y caras en una topología espacial.
- Modelo de datos (de red) para trabajar con elementos modelados, como nodos o enlaces, en una red.
- GeoRaster, una función que permite almacenar, indexar, consultar, y analizar datos GeoRaster.

2.3.2 Estudio general de Oracle Spatial.

Como se ha avanzado en el apartado anterior, Oracle Spatial permite almacenar, analizar y utilizar datos de una base de datos. Será a lo largo de este apartado cuando se vea en mayor profundidad la arquitectura de este paquete así como algunas de las posibilidades y tipos de datos que brinda.

En primer lugar, se describen los componentes que forman el paquete Oracle Spatial, es decir, su arquitectura. Oracle Spatial se distribuye en dos niveles: la propia base de datos Oracle y el servidor de aplicaciones [35].

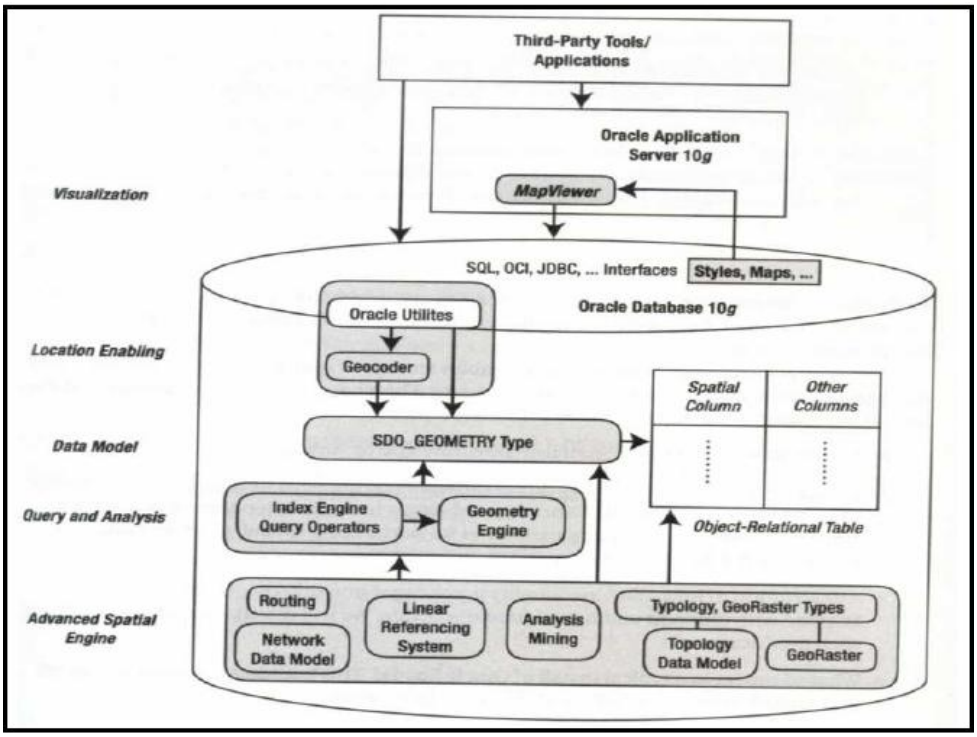


Figura 1. Componentes de la tecnología de Oracle Spatial

Los componentes de la base de datos son los siguientes:

- Data Model: es la estructura jerárquica compuesta por elementos, objetos geométricos y capas. Las capas están compuestas por objetos geométricos y éstos a su vez se forman de elementos. Los elementos son el nivel más básico de esta estructura, y pueden ser puntos, líneas o polígonos, conformados por sus dimensiones correspondientes. Los objetos geométricos son la representación de estos elementos, y las capas son una colección de objetos geométricos. Además de esta estructura cabe destacar el sistema de coordenadas o referencias (Coordinate System) que asigna las coordenadas correspondientes a una localización concreta. Por último cabe destacar otro aspecto importante del Data Model que es la tolerancia (Tolerance) que se relaciona con el nivel de precisión de los datos. Refleja la distancia que pueden separar a dos puntos para considerarse el mismo. A lo largo del presente documento se verá algún ejemplo y se explicará más en detalle este aspecto, aunque básicamente consiste en darle valor al atributo SDO_TOLERANCE (incluido en la vista USER_SDO_GEOM_METADATA). Si la unidad que rige el sistema referencial (coordenadas) es el metro (datos geodésicos: longitud, latitud), el

valor de tolerancia estará directamente relacionado con este valor del sistema referencial, de forma que si la tolerancia tiene valor 100, indica que si hay dos objetos separados por menos de 100 metros en el sistema de coordenadas, éstos serán considerados iguales.

- Location Enabling: gracias a los tipos de datos espaciales, se pueden crear tablas con columnas que almacenen datos de ese tipo. Para este proyecto, por ejemplo, se crea una tabla para almacenar los nodos, cuya forma geométrica será el punto, en los que se deberá indicar en la columna SDO_GEOMETRY qué tipo de dato es el nodo que se va a guardar y las coordenadas en las que se sitúa. Este tipo de inserciones se puede realizar también gracias a las utilidades estándar de Oracle.
- Spatial Query and Analysis: la base de datos Oracle cuenta con un motor de geometría y un motor de índice que permite realizar consultas y análisis de los datos espaciales. Para enriquecer las operaciones de búsqueda más comunes, como la clausula SELECT, o para la creación de índices, el componente Oracle Spatial cuenta con una serie de operadores que permiten refinar el resultado en función del interés del usuario (estos operadores se incluyen en la vista USER_SDO_GEOM_METADATA).

  Por ejemplo, el operador SDO_DISTANCE permite calcular la distancia que separa a dos objetos, el operador SDO_AREA calcula el área de un polígono, SDO_CENTROID devuelve el centro de un polígono, SDO_LENGTH calcula la longitud del perímetro de un polígono, y así un sinfín de operadores que dotan a Oracle Spatial de una gran riqueza para manejar datos espaciales.
- Visualization: permite visualizar los resultados de las consultas espaciales. Esto es llevado a cabo por el componente MapViewer, que genera mapas de visualización en función de esos datos.
- Advanced Spatial Engine: contiene subcomponentes necesarios para el análisis y manejo de datos espaciales que se requieren en las aplicaciones GIS tradicionales. Entre estas funcionalidades destacan el almacenamiento de redes dentro de la base de datos Oracle, gracias al Modelo de Datos de Red, o el almacenamiento y recuperación de imágenes georeferenciadas, gracias al subcomponente GeoRaster.

Vistas de Metadatos Geométricos (Espaciales)

Los metadatos geométricos o espaciales que describen las dimensiones, tolerancia de las mismas, etc., se almacenan en una tabla global con permiso MDSYS. En función del esquema asociado, el usuario Spatial tiene disponible la vista USER_SDO_GEOM_METADATA.
La definición de la vista se muestra en el Ejemplo 1.

```
(
TABLE_NAME      VARCHAR2(32),
COLUMN_NAME     VARCHAR2(32),
DIMINFO         SDO_DIM_ARRAY,
SRID            NUMBER
);
```

Ejemplo 1. Definición de la vista USER_SDO_GEOM_METADATA

Como se puede observar, la vista está compuesta por 4 atributos o columnas. La columna TABLE_NAME contiene el nombre de la tabla. Esta columna es de tipo VARCHAR2. La columna COLUMN_NAME contiene el nombre de la columna, también es de tipo VARCHAR2. Estas dos columnas, en definitiva, identifican la capa espacial que se desea guardar.

La columna DIMINFO es un array de longitud variable. Especifica la información sobre cada dimensión de la capa. Esta columna es de tipo SDO_DIM_ARRAY. Éste se define en el Ejemplo 2.

```
Create Type SDO_DIM_ARRAY as VARRAY(4) of SDO_DIM_ELEMENT;
```

Ejemplo 2. Definición de la columna DIMINFO

A su vez, el tipo SDO_DIM_ELEMENT se define en el Ejemplo 3.

```
Create Type SDO_DIM_ELEMENT as OBJECT (
SDO_DIMNAME VARCHAR2 (64),
SDO_LB NUMBER,
SDO_UB NUMBER,
SDO_TOLERANCE NUMBER);
```

Ejemplo 3. Definición del tipo SDO_DIM_ELEMENT

Como se puede observar, SDO_TOLERANCE es un atributo de DIMINFO. Como ya se ha indicado, la tolerancia define un nivel de precisión entre datos espaciales. Como norma general, el valor de la tolerancia se debe establecer como la menor distancia que se pueda distinguir entre dos objetos. Normalmente, esta distancia corresponde a la mitad de la diferencia entre los valores de dos coordenadas.

Por último, la vista USER_SDO_GEOM_METADATA tiene un atributo SRID. Este atributo especifica el valor del sistema de coordenadas para todos los datos espaciales. Normalmente su valor es nulo, que indica que el sistema de referencia asociado a los objetos geométricos es el sistema cartesiano. No obstante, algunos de los valores más utilizados para este atributo son 8307, conocido como "Longitud / Latitud (WGS 84)", que es el sistema más usado para posicionamiento GPS, y también el valor 8199, llamado "Longitud / Latitud (Arc 1950). El sistema de coordenadas podría ser geodésico, de proyección o local.
Un ejemplo de inserción en esta vista se muestra en el Ejemplo 4.

```
INSERT INTO user_sdo_geom_metadata
(TABLE_NAME,
COLUMN_NAME,
DIMINFO,
SRID)
VALUES (
'cola_markets',
'shape',
SDO_DIM_ARRAY( -- 20X20 grid
SDO_DIM_ELEMENT('X', 0, 20, 0.05),
SDO_DIM_ELEMENT('Y', 0, 20, 0.05)
),
NULL -- SRID
);
```

Ejemplo 4. Inserción en la tabla USER_SDO_GEOM_METADATA

En este ejemplo se crea la tupla con nombre "cola_markets", cuyo nombre de columna será "shape". Esta tupla estará formada por un array de 20x20, tal y como indican los valores de SDO_DIM_ELEMENT, en los que a su vez se fija una tolerancia (precisión) de 0.05. El valor de SRID se establece como NULL, de forma que se asocia al sistema de coordenadas cartesianas.

El tipo de datos SDO_GEOMETRY

Siguiendo un orden jerárquico se procede a estudiar el tipo de datos SDO_GEOMETRY, ya que es el encargado de incluir las geometrías en la capa correspondiente, tal y como se verá con algunos ejemplos más adelante.

Este tipo de datos permite almacenar variedad de datos espaciales, entre los que destacan los puntos, líneas, polígonos y geometrías complejas. Algunos ejemplos se pueden ver en la figura x. Este proyecto se centrará, sobre todo, en los tres primeros.

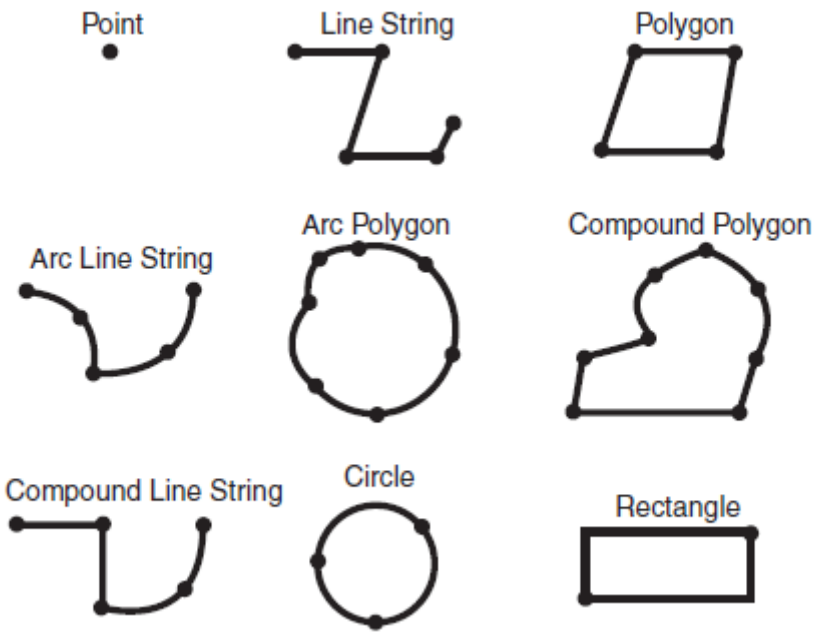


Figura 2. Ejemplo de geometrías posibles a guardar en SDO_GEOMETRY

Antes de analizar algunas de las geometrías es preciso estudiar la estructura del objeto SDO_GEOMETRY, tal y como se muestra en el Ejemplo 5.

```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Ejemplo 5. Estructura del objeto SDO_GEOMETRY

A continuación se detallan cada uno de los atributos que forman este tipo de datos:

- SDO_GTYPE: especifica el tipo de forma geométrica del objeto. Este atributo es un valor de 4 dígitos con formato *dltt* donde la "d" identifica el número de dimensiones (2, 3 o 4) de forma que si, por ejemplo, es bidimensional cada vértice tiene dos coordenadas. La "l" se utiliza con referencias lineales, tales como aplicaciones de rutas de transporte, tuberías, etc., siendo 0 su valor por defecto que es el que se utiliza en nuestro sistema. Por último, "tt" identifica el tipo de geometría (con valores desde 00 a 07, mientras que los valores desde 08

a 99 están reservados para futuros casos). El valor *dl00* es un tipo no interpretado, *dl01* es el valor para el punto, es decir, la forma geometría contendría un punto, *dl02* para líneas o curvas, *dl03* para un polígono, *dl04* para una colección, *dl05* para un multipunto, *dl06* para una multilínea o una multicurva y *dl07* para un multipolígono.

- SDO_SRID: especifica el sistema de coordenadas asociado a la geometría. Como se ha indicado anteriormente, si el valor de este atributo es nulo, se asocia al sistema de ejes cartesianos. En caso contrario, es decir, el valor no es nulo, éste debe contener un valor de la tabla SDO_COORD_REF_SYS. Además este mismo valor debe insertarse en la columna SRID de la vista USER_SDO_GEOM_METADATA.
- SDO_POINT: este atributo se utiliza para especificar las coordenadas en las que se sitúa una geometría de tipo punto. Es de tipo SDO_POINT_TYPE, cuya estructura contiene tres atributos de tipo NUMBER que representan las coordenadas del punto. Más adelante se verá un ejemplo.
- SDO_ELEM_INFO: sirve para almacenar objetos espaciales más complejos. Es un array de números (tipo SDO_ELEM_INFO_ARRAY) y su funcionamiento es el siguiente:

  Cada tres números consecutivos del array se agrupan internamente de forma que representen un objeto geométrico o una parte de él, como por ejemplo un arco en una determinada figura. Este "triplete" tiene el formato <SDO_STARTING_OFFSET, SDO_ETYPE, SDO_INTERPRETATION>. El offset representa el comienzo del array SDO_ORDINATES donde están almacenadas las coordenadas de los elementos, es decir, hace la función de índice en el array para conocer las coordenadas de cada elemento que forma el objeto. Su valor siempre suele ser un uno. Los valores de Etype y de Interpretation representan el objeto asociado, es decir, el objeto que se quiere representar (un punto, una línea, un polígono), así como la representación de sus límites.

  Por ejemplo, si el valor de SDO_ETYPE y de SDO_INTERPRETATION es 1 (con lo que el triplete sería <1, 1, 1>) se desea representar un punto. En el caso de un triplete <1, 2, 1> se estarían representando una cadena de líneas. Si este triplete fuera <1, 2, 2> esta cadena de líneas estarían unidas por

arcos.

Para representar un polígono simple, conectado por líneas rectas, se utiliza el triplete <1, 1003 o 2003, 1>. Al hilo de este ejemplo, si el triplete fuera <1, 1003 o 2003, 2>, el polígono estaría unido por arcos; si el triplete fuera <1, 1003 o 2003, 3> el polígono sería rectángulo y para el triplete <1, 1003 o 2003, 4> sería un polígono circular.

En definitiva, son los valores del triplete descriptor los que marcan la representación de los objetos espaciales.

- SDO_ORDINATES: este atributo es un array de números que almacena el valor de las coordenadas de los vértices que limitan un objeto. Por ejemplo, si se quiere representar una línea que une dos vértices A y B, cuyas coordenadas son (Ax, Ay) y (Bx, By), el valor de este atributo SDO_ORDINATES contendrá de forma ordenada estas cuatro coordenadas.

Ejemplos de geometrías

Con el objetivo de aclarar los conceptos anteriormente expuestos, se detallan a continuación algunos ejemplos de geometrías.

Punto

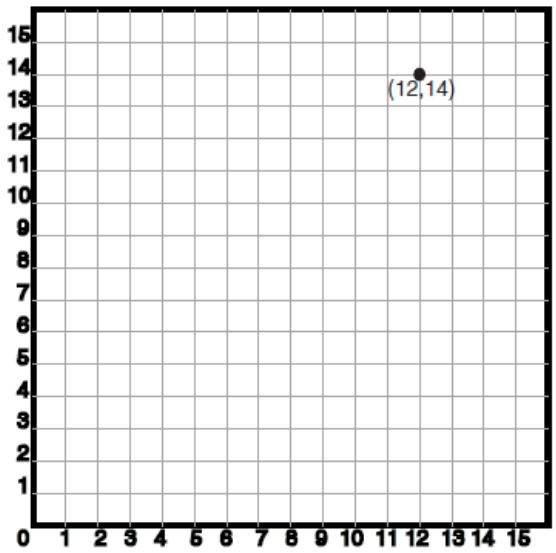


Figura 3. Representación de un punto en un sistema de coordenadas

Para representar un punto en el sistema de coordenadas (siguiendo como ejemplo el de la figura 3) los atributos del tipo SDO_GEOMETRY deben contener los valores correctos.

- SDO_GTYPE = 2001. Es el valor que debe contener este atributo. El dígito 2 indica que el objeto, el punto, es de dos dimensiones y el 1 indica que hay un único punto.
- SDO_SRID = NULL, es decir, sistema de coordenadas cartesiano.
- SDO_POINT = SDO_POINT_TYPE (12, 14, NULL). Como solo tiene dos coordenadas, éstas son las que se incluyen en este atributo. La coordenada X tiene valor 12 y la coordenada Y valor 14.
- SDO_ELEM_INFO y SDO_ORDINATES tienen valor nulo pues para la geometría de tipo punto no es necesario incluir los vértices que forman los límites del objeto.

El ejemplo completo en SQL se muestra en el Ejemplo 6.

```
INSERT INTO cola_markets VALUES(
90,
'point_only',
SDO_GEOMETRY(
2001,
NULL,
SDO_POINT_TYPE(12, 14, NULL),
NULL,
NULL));
```

Ejemplo 6. Inserción de un punto

Como se indicaba anteriormente, esta inserción almacena el punto en la tabla "cola_markets" creada previamente.

Rectángulo

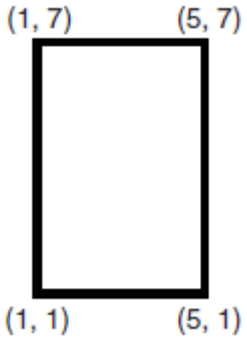


Figura 4. Representación de un rectángulo

Al igual que para el caso del Punto, para representar un rectángulo con sus coordenadas correspondientes (siguiendo como ejemplo el de la figura 4), a continuación se detallan los valores del objeto SDO_GEOMETRY para esta geometría:

- SDO_GTYPE = 2003. Siguiendo el formato dltt, el 2 indica que la figura es de dos dimensiones (dos coordenadas por vértice) y el valor 3 indica que es un polígono.
- SDO_SRID = NULL, es decir, sistema de coordenadas cartesiano.
- SDO_POINT = NULL. Tiene valor nulo porque las geometrías distintas del tipo punto no usan este atributo.
- SDO_ELEM_INFO = (1, 1003, 3). El dígito 3 al final del triplete, es decir, el valor de SDO_INTERPRETATION, indica que es un rectángulo.
- SDO_ORDINATES = (1, 1, 5, 7). Al ser un rectángulo solo es necesario especificar dos ordenadas en este atributo, siendo éstas el vértice inferior izquierdo y vértice superior derecho.

El ejemplo completo en SQL se muestra en el Ejemplo 7.

```
INSERT INTO cola_markets VALUES(
1,
'cola_a',
SDO_GEOMETRY(
2003, -- polígono de dos dimensiones
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1, 1003, 3),    -- un rectángulo (1003 = exterior)
SDO_ORDINATE_ARRAY(1, 1,  5, 7)    -- sólo dos puntos son necesario para definir -- el rectángulo (inferior izquierda y superior derecha) con coordenadas cartesianas
)
);
```

Ejemplo 7. Inserción de un rectángulo

Cadena de líneas compuesta (líneas rectas y arcos)

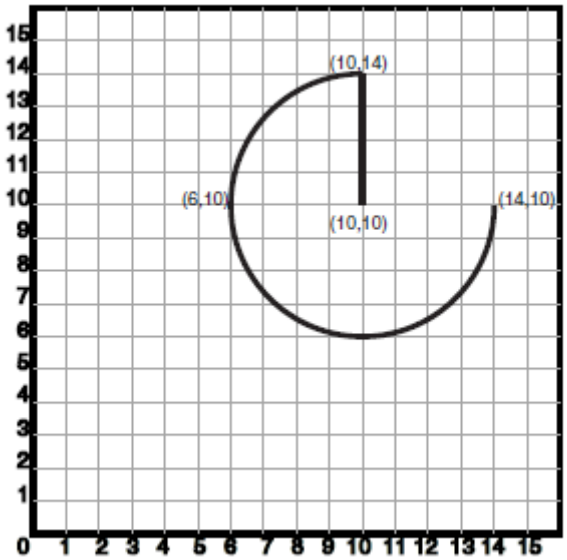


Figura 5. Representación de una cadena de líneas compuesta

Para representar este objeto geométrico en el sistema de coordenadas (siguiendo como ejemplo el de la figura 5), a continuación se detallan los valores de los distintos atributos del objeto SDO_GEOMETRY para esta geometría de ejemplo:

- SDO_GTYPE = 2002. Siguiendo el formato dltt, el 2 indica que la figura es de dos dimensiones (dos coordenadas por vértice) y el valor 2 final indica que hay dos o más segmentos.

- SDO_SRID = NULL, es decir, sistema de coordenadas cartesiano.
- SDO_POINT = NULL. Tiene valor nulo al no ser un punto.
- SDO_ELEM_INFO = (1, 4, 2, 1, 2, 1, 3, 2, 2). Hay tres tripletes descriptores. El primer triplete (1, 4, 2) indica que el objeto es una cadena de líneas formada por dos subelementos cadena de línea. El segundo triplete (1, 2, 1) indica que los extremos de la línea (primer subelemento) están conectados por líneas rectas. Tiene como comienzo el valor del punto en el offset 1. El punto final de esta línea está determinado por el comienzo del offset del triplete siguiente, es decir, el valor 3. El tercer triplete (3, 2, 2) indica que la segunda línea tiene forma de arco, con valor del offset 3. El punto final de esta línea viene determinado por el valor offset del siguiente triplete o, en este caso, por la longitud actual del array SDO_ORDINATES.
- SDO_ORDINATES = (10, 10, 10, 14, 6, 10, 14, 10).

El ejemplo completo en SQL se muestra en el Ejemplo 8.

```
INSERT INTO cola_markets VALUES(
1,
'cola_a',
SDO_GEOMETRY(
2003, -- polígono de dos dimensiones
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1003,3),    -- un rectángulo (1003 = exterior)
SDO_ORDINATE_ARRAY(1,1, 5,7)    -- sólo dos puntos son necesario para definir
-- el rectángulo (inferior izquierda y superior derecha) con coordenadas cartesianas
)
);
```

Ejemplo 8. Inserción de un objeto de cadena de líneas compuesta

Polígono compuesto

Para finalizar con esta serie de ejemplos de geometrías, a continuación se detalla un ejemplo de polígono compuesto, tal y como se muestra en la figura 6.

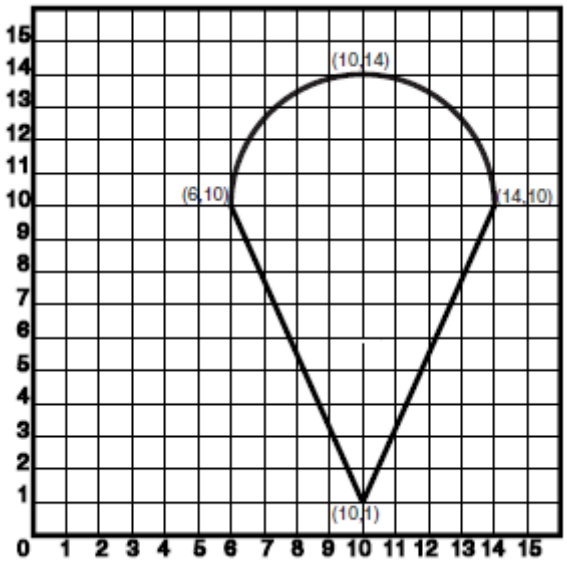


Figura 6. Representación de un polígono compuesto

Los valores de los atributos del objeto SDO_GEOMETRY para esta geometría de ejemplo serían los siguientes:

- SDO_GTYPE = 2003. Siguiendo el formato dltt, el 2 indica que la figura es de dos dimensiones (dos coordenadas por vértice) y el valor 3 final indica que el objeto es un polígono.
- SDO_SRID = NULL, es decir, sistema de coordenadas cartesiano.
- SDO_POINT = NULL. Tiene valor nulo al no ser un punto.
- SDO_ELEM_INFO = (1, 1005, 2, 1, 2, 1, 5, 2, 2). Hay tres tripletes descriptores.

  El primer triplete (1, 1005, 2) indica que la geometría es un polígono compuesto. El segundo triplete (1, 2, 1) indica que los extremos de la línea (primer subelemento) están conectados por líneas rectas. Tiene offset 1. El punto final de esta línea está determinado por el comienzo del offset del triplete siguiente, es decir, el valor 5.

El tercer triplete (5, 2, 2) indica que el siguiente subelemento está conectado por arcos, es decir, que la segunda línea tiene forma de arco, con valor del offset 5. El punto final de esta línea viene determinado por el valor offset del siguiente triplete o, en este caso, por la longitud actual del array SDO_ORDINATES.

- SDO_ORDINATES = (6, 10, 10, 1, 14, 10, 10, 14, 6, 10).

El ejemplo completo en SQL se muestra en el Ejemplo 9.

```
INSERT INTO cola_markets VALUES(
12,
'compound_polygon',
SDO_GEOMETRY(
2003, -- polígono de dos dimensiones
NULL,
NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 5,2,2), -- polígono compuesto
SDO_ORDINATE_ARRAY(6,10, 10,1, 14,10, 10,14, 6,10)
)
);
```

Ejemplo 9. Inserción de un polígono compuesto

2.4. Entorno de trabajo

En este apartado se analiza y se detalla la tecnología y herramientas software utilizadas para implementar el sistema, tales como el lenguaje de programación JAVA junto a la herramienta NetBeans, el lenguaje SQL para los procesos relacionados con la base de datos, y el lenguaje de modelado UML.

2.4.1 Lenguaje de programación Java y plataforma NetBeans

Para la implementación de software se ha utilizado NetBeans, una plataforma de código abierto que cuenta con un gran éxito y una comunidad de usuarios en constante crecimiento [36]. Sun Microsystems fundó el proyecto NetBeans en junio del año 2.000 y continúa siendo el patrocinador principal junto con cerca de 100 socios de todo el mundo. El proyecto NetBeans facilita una herramienta llamada IDE NetBeans, cuya versión utilizada ha sido la 6.8 (última versión estable en el momento del desarrollo del proyecto). Esta herramienta IDE no es más que un entorno de desarrollo integrado (IDE

es el acrónimo en inglés de Integrated Development Environment) para programadores, con el objetivo de que éstos puedan escribir, compilar, depurar y ejecutar los programas de una manera cómoda. Una de las ventajas de esta herramienta es que aunque está escrita en Java puede servir para cualquier otro lenguaje.

Para este proyecto se ha decidido contar con esta herramienta por varios motivos:

- Como ya se ha indicado, el proyecto NetBeans es de código abierto, por lo que la herramienta IDE NetBeans es un producto gratuito y sin restricciones de uso. Es más, existen varios "paquetes" de instalación diferentes en función de las necesidades del programador.
- Otra de las ventajas del producto es la gran comunidad de usuarios que respalda el proyecto y colabora con él. Como la herramienta está basada en módulos (la modularidad es otra de sus grandes ventajas) existe una cantidad importante de módulos que extienden el proyecto.
- Una ventaja más a destacar es la facilidad para elaborar interfaces gráficas. Para este proyecto es imprescindible la elaboración de pantallas y elementos que permitan interaccionar al usuario. Aunque se valoró la opción de utilizar Eclipse, que es otro entorno de desarrollo integrado de código abierto similar [37], ha sido el aspecto de la comodidad para crear interfaces gráficas la que ha decantado la balanza a favor de IDE NetBeans.

Esta herramienta permite la implementación de código escrito en lenguaje Java, del que se ha utilizado la versión 6 "JSE 6". Java, al ser el código en el que está escrito NetBeans, mantiene las características de libertad y uso que le hacen ser un lenguaje muy utilizado. Java además es un lenguaje independiente de la plataforma que lo interpreta ya que es la máquina virtual de Java (JVM, acrónimo en inglés de Java Virtual Machine) instalada en el equipo la que comunica el sistema operativo con la aplicación Java en cuestión para que ésta sea interpretada y ejecutada correctamente.

2.4.2 Lenguaje SQL

El SQL es el lenguaje estándar ANSI/ISO de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo: sólo hay que indicar que se quiere

hacer [38]. Este lenguaje se ha usado para gestionar y realizar operaciones en la base de datos Oracle 10G Express Edition que se ha utilizada. Esta base de datos, instalada en local, permite ejecutar una ventana para introducir los comandos que se precisen, que es la herramienta que se ha utilizado para conectar con la base de datos así como para gestionar la creación de tablas, inserción de registros, borrados, updates, etc.

2.4.3 Lenguaje UML

La elaboración del diagrama de clases y los casos de uso se basa en el lenguaje UML, que son las siglas de Unified Modeling Language, es decir, Lenguaje Unificado de Modelado. UML está considerado como el lenguaje de modelado de sistemas más conocido y utilizado en la actualidad, promovido por el OMG (Object Management Group) que se encarga de cuidar algunos estándares de tecnologías orientadas a objetos. UML es un lenguaje para describir procesos o métodos, y en definitiva para especificar, construir, y documentar un sistema. Es importante resaltar que UML no es un lenguaje de programación, como lo puede ser la programación estructurada u orientada a objetos, sino que muestra una descripción del sistema a través de diagramas.

3. Análisis y Diseño

Antes de desarrollar el sistema, se ha realizado un análisis exhaustivo de los componentes hardware y software que participan en él para, posteriormente, realizar un diseño de las distintas herramientas que se han utilizado con el fin de alcanzar los objetivos propuestos.

3.1. Especificación de requisitos software

Para conocer el comportamiento de la aplicación desarrollada, previamente se ha de conocer qué debe hacer la aplicación, cuál es su alcance, y las restricciones con las que se cuenta. Por ello, a la hora de comenzar un proyecto, una de las primeras tareas es la de reunirse con el cliente con el fin de conocer sus necesidades y sus ideas respecto al objeto del proyecto, es decir, la toma de requisitos.

En este caso, aún siendo un escenario diferente al supuesto anterior por no realizar este proyecto para un cliente concreto, se han de analizar los requisitos que rigen la elaboración del proyecto con el objetivo de enmarcar el diseño y la posterior implementación de la aplicación.
Para ello, se dividen los requisitos en tres tipos, requisitos funcionales, requisitos no funcionales y requisitos de interfaz.

3.1.1. Requisitos funcionales

En este apartado se estudian los requisitos que están relacionados con los casos de uso detallados en el apartado 3.2.

Para detallar cada uno de ellos se utiliza la siguiente nomenclatura: RFXX donde RF son las siglas abreviadas de Requisito Funcional y XX el número del requisito.

Por tanto, los requisitos funcionales son los siguientes:

- RF01: El sistema debe permitir el acceso de dos usuarios de perfil diferente (un usuario Administrador y un usuario Consultor).

- RF02: El sistema debe verificar que la comunicación establecida entre el lector RFID y las etiquetas es correcta, y que éstas devuelven la información correspondiente.

- RF03: El sistema debe almacenar la información que introduzca el usuario, de forma que los usuarios de perfil Administrador podrán crear y borrar Puntos, y los usuarios de perfil Consultor podrán crear una red, es decir, crear y borrar Nodos y Enlaces.

- RF04: El sistema, en base al requisito funcional RF03, debe poder almacenar objetos geo-espaciales.

- RF05: Los elementos almacenados deberán estar posicionados en base a un eje cartesiano de dos dimensiones (x,y).

- RF06: Debe permitirse la creación de una red, la cual se dibujará sobre un tablero, en base al requisito RF05, en el que se dispondrán los distintos nodos y enlaces que se añadan.

- RF07: Un requerimiento de calidad es la modularidad del sistema, es decir, éste debe estar desarrollado mediante módulos independientes integrados, que permita evolucionar módulos por separado.

- RF08: Escalabilidad. El sistema debe estar desarrollado de tal forma que pueda crecer con facilidad ante diferentes dimensiones del escenario (tablero).

- RF09: Aprendizaje del programa: se crea un manual de usuario, desde el punto de vista práctico, para su fácil comprensión.

Además, en este apartado, cabe destacar los siguientes requisitos de interfaz:

- RF10: Se debe estudiar la disposición de los componentes en pantalla, tales como cuadros de texto, botones, barra de menú, desplegables, etc.

- RF11: A su vez, el botón de desconexión o salida del programa debe estar disponible en todo momento.

3.1.2. Requisitos no funcionales

En este apartado se agrupan los requisitos de recursos, rendimiento, mantenimiento, portabilidad, documentación y seguridad, entre otros.

Para detallar cada uno de ellos se utiliza la siguiente nomenclatura: RNFXX donde RNF son las siglas abreviadas de Requisito no funcional y XX el número del requisito. Previo al detalle de cada requisito se explica el tipo del que forma parte en función de las agrupaciones citadas anteriormente.
Por tanto, los requisitos no funcionales son los siguientes:

1. Los **requisitos de recursos** especifican las características de los equipos y dispositivos participantes en la infraestructura del sistema.

   - RNF01: Se precisan etiquetas RFID y un lector de etiquetas RFID con un rango de lectura de varios metros. La aplicación no precisa un PC demasiado potente pues con 1 GB de RAM es suficiente. Por último, la resolución de pantalla óptima es de 1024x768 píxeles.

   - RNF02: Se precisa un convertidor de puerto RS-232 a puerto USB con el objetivo de poder utilizar el sistema en cualquier equipo ya que, actualmente, la mayoría de los equipos modernos no cuentan con un puerto serial RS-232.

2. Los **requisitos de rendimiento** especifican cálculos temporales como, entre otros, el acceso a la base de datos.

   - RNF03: El tiempo de arranque de la aplicación debe ser eficiente.

   - RNF04: El tiempo de respuesta de la Base de Datos debe ser eficiente.

   - RNF05: El tiempo de carga de los elementos en el tablero debe ser prácticamente inmediato.

3. Los **requisitos de mantenimiento** especifican aquellos que la aplicación debe satisfacer con el objetivo de adaptarse correctamente al desarrollo de posibles futuras mejoras sin necesidad de un coste de recursos excesivo.

- RNF06: El sistema debe contar con la posibilidad de ser ampliado en un futuro, por lo que la implementación deberá estar comentada y documentada, así como cumplir los estándares de programación.

4. Los **requisitos de portabilidad** especifican los cambios precisos para que el software funcione correctamente en otras máquinas y sistemas, con diferentes sistemas operativos, resoluciones de pantalla, etc. Es decir, especifican la adaptación del software al entorno.

   - RNF07: La máquina en la que corra la aplicación debe contar con un puerto USB para conectar el convertidor que a su vez se conectará con el lector RFID. No obstante, si la máquina contiene conexión por puerto serial RS-232 no es necesario la utilización del convertidor de puerto, y el lector se podrá conectar directamente al PC. En cuanto al software, la aplicación funcionará correctamente en cualquier máquina que tenga instalada el entorno JSE versión 6.

5. Los **requisitos de documentación** describen las tareas que se deben realizar con el objetivo de documentar el proyecto completamente, incluyendo el manual de usuario de la aplicación.

   - RNF08: Desde el punto de vista del programador es imprescindible la realización de un manual de usuario así como de un documento en el que se facilite el diagrama de clases, con cada una de ellas detallada, con el objetivo de que futuros desarrolladores de la aplicación puedan adaptarse lo más pronto posible a la misma.

6. Los **requisitos de seguridad** describen el desarrollo de medidas que previenen ataques externos, como por ejemplo autentificaciones no seguras a la aplicación.

   - RNF09: Para utilizar las diversas funcionalidades que la aplicación facilita se debe realizar una conexión y validación de usuario, por lo que se creará una ventana de diálogo a tal efecto.

3.2. Casos de Uso

La especificación de casos de uso [39] proporciona la funcionalidad requerida para el sistema y los distintos estadios y situaciones en las que los usuarios interaccionan con él.

En este caso, en el sistema interactúan dos actores: el actor Administrador, encargado de modelar el escenario creando puntos, y el actor Consultor, encargado de crear la red y visualizar el posicionamiento en tiempo real. Los diagramas de uso, en función del actor (perfil), se muestran en las figuras 7 y 8.

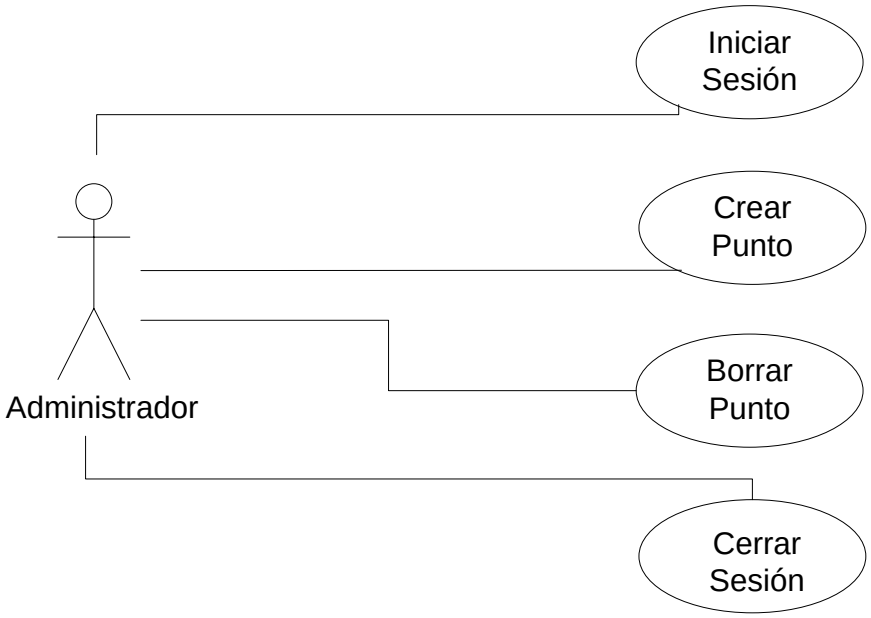


Figura 7. Diagrama de casos de uso del actor Administrador

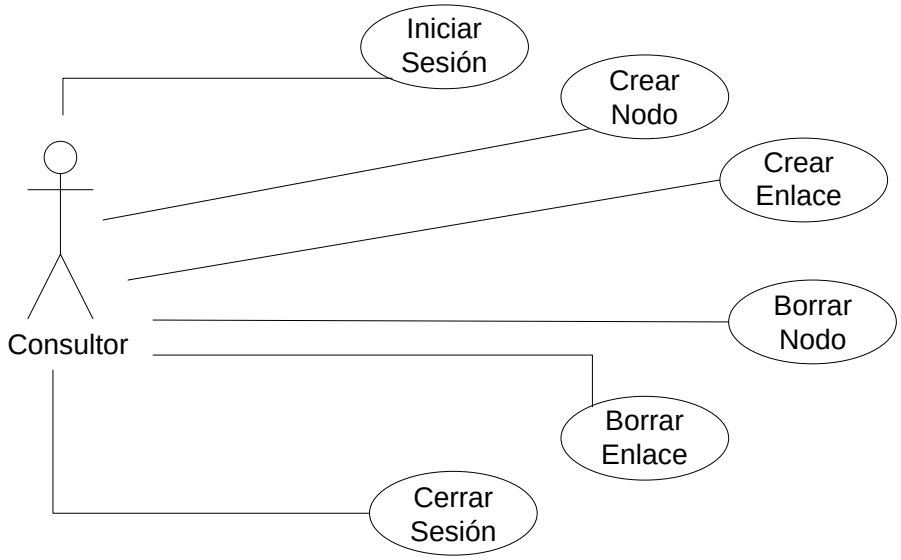


Figura 8. Diagrama de casos de uso del perfil Consultor

Para describir cada uno de los casos de uso se utilizarán tablas de especificación textual como la que se muestra en la tabla 1.

Nombre	**Frase verbal descriptiva**
Descripción	Descripción del caso de uso.
Actores	Actores que interaccionan con el sistema participando en este caso de uso.
Objetivo	Finalidad o servicio requerido.
Precondiciones	Descripción del estado del sistema antes de la ejecución del caso de uso.
Postcondiciones	Descripción del estado del sistema después de la ejecución del caso de uso.
Escenario básico	Secuencia de acciones principales de la interacción en el escenario básico. Información intercambiada.

Tabla 1. Especificación textual para un caso de uso

Por tanto, a continuación se procede a describir cada caso de uso:

El caso de uso de inicio de sesión se muestra en la Tabla 2.

Nombre	**Iniciar Sesión**
Descripción	Inicio de sesión por parte del usuario, ya sea Administrador o Consultor.
Actores	Administrador y Consultor.
Objetivo	Restricción de acceso al sistema. Seguridad de acceso al mismo.
Precondiciones	Pantalla de bienvenida de la aplicación y acceso a la opción Conectar del menú. Pantalla de introducción de datos de usuario.
Postcondiciones	Proporciona el perfil correspondiente al usuario, permitiendo acceder a las herramientas asociadas al mismo.
Escenario básico	- El usuario pulsa el botón Archivo del menú y posteriormente la opción Conectar. - Se muestra una ventana de diálogo (emergente) con los campos Nombre y Contraseña para que el usuario introduzca sus datos. - El usuario introduce sus datos y pulsa Aceptar.

Tabla 2. Caso de uso Iniciar Sesión

El caso de uso de cierre de sesión se muestra en la Tabla 3.

Nombre	**Cerrar Sesión**
Descripción	Cierre de la sesión por parte del usuario, ya sea Administrador o Consultor.
Actores	Administrador y Consultor
Objetivo	Cerrar sesión.
Precondiciones	Sólo se puede cerrar sesión si se ha iniciado antes, de forma que el usuario se encontrará en el panel correspondiente según su perfil.

Postcondiciones	Vuelta a la pantalla de inicio o bienvenida de la aplicación.
Escenario básico	- Una vez iniciada la sesión, independientemente del perfil del usuario, se puede cerrar la sesión gracias al menú del panel. - Se debe pulsar en el menú Archivo y posteriormente la opción Desconectar.

Tabla 3. Caso de uso Cerrar Sesión

El caso de uso de creación de un punto se muestra en la Tabla 4.

Nombre	**Crear Punto**
Descripción	Creación de un punto
Actores	Administrador
Objetivo	Crear un punto para modelar el tablero
Precondiciones	Esta opción estará disponible si previamente el usuario ha validado sus datos en el sistema, siendo éste de perfil Administrador.
Postcondiciones	Punto creado
Escenario básico	- Una vez iniciada la sesión del usuario perfil Administrador, se podrán crear puntos. - Se pulsa la opción Archivo – Crear Punto. - Se rellenan los datos correspondientes (coordenadas del punto y descripción si procede) y se pulsa Aceptar para crear el punto, que conlleva el almacenamiento de estos datos en su tabla correspondiente de la base de datos.

Tabla 4. Caso de uso Crear Punto

El caso de uso de borrado de un punto se muestra en la Tabla 5.

Nombre	**Borrar Punto**
Descripción	Borrado de un punto

Actores	Administrador
Objetivo	Eliminar un punto previamente almacenado en la base de datos.
Precondiciones	Esta opción estará disponible si previamente el usuario ha validado sus datos en el sistema, siendo éste de perfil Administrador. Además, deben existir puntos creados.
Postcondiciones	Punto eliminado.
Escenario básico	- Una vez iniciada la sesión del usuario perfil Administrador, se podrán eliminar puntos. - Se pulsa la opción Archivo – Borrar Punto. - Se elige el punto que se quiere borrar de la lista desplegable.

Tabla 5. Caso de uso Borrar Punto

El caso de uso de creación de un nodo se muestra en la Tabla 6.

Nombre	**Crear Nodo**
Descripción	Creación de un nodo
Actores	Consultor
Objetivo	Crear un nodo para posicionar a futuros usuarios del sistema
Precondiciones	Esta opción estará disponible si previamente el usuario ha validado sus datos en el sistema, siendo éste de perfil Consultor.
Postcondiciones	Nodo creado y dibujado en el tablero.

Escenario básico	- Una vez iniciada la sesión del usuario perfil Consultor, se podrán crear nodos. - Para ello, se ha de acceder al panel de creación de Red (Archivo – Crear Red). - Se rellenan los datos correspondientes (descripción si procede) y se pulsa Aceptar para crear el Nodo, que conlleva el almacenamiento de estos datos en su tabla correspondiente de la base de datos.

Tabla 6. Caso de uso Crear Nodo

El caso de uso de borrado de un nodo se muestra en la Tabla 7.

Nombre	**Borrar Nodo**
Descripción	Borrado de un nodo
Actores	Consultor
Objetivo	Eliminar un nodo previamente almacenado en la base de datos.
Precondiciones	Esta opción estará disponible si previamente el usuario ha validado sus datos en el sistema, siendo éste de perfil Consultor. Además, deben existir nodos creados.
Postcondiciones	Nodo eliminado de la base de datos y del tablero.
Escenario básico	- Una vez iniciada la sesión del usuario perfil Consultor, se podrán borrar nodos. - Para ello, se ha de acceder al panel de creación de Red (Archivo – Crear Red). - Se introduce el ID del nodo a borrar y se pulsa Aceptar.

Tabla 7. Caso de uso Borrar Nodo

El caso de uso de creación de un enlace se muestra en la Tabla 8.

Nombre	**Crear Enlace**
Descripción	Creación de un enlace entre nodos
Actores	Consultor
Objetivo	Crear un enlace, es decir, unir dos nodos.
Precondiciones	Esta opción estará disponible si previamente el usuario ha validado sus datos en el sistema, siendo éste de perfil Consultor.
Postcondiciones	Enlace creado y dibujado en el tablero.
Escenario básico	- Una vez iniciada la sesión del usuario perfil Consultor, se podrán crear enlaces. - Para ello, se ha de acceder al panel de creación de Red (Archivo – Crear Red). - Se indican los dos nodos que formarán el enlace y se pulsa el botón Aceptar. Los datos correspondientes se almacenan en las tablas correspondientes de la base de datos.

Tabla 8. Caso de uso Crear Enlace

El caso de uso de borrado de un enlace se muestra en la Tabla 9.

Nombre	**Borrar Enlace**
Descripción	Borrado de un enlace
Actores	Consultor
Objetivo	Eliminar un enlace previamente almacenado en la base de datos.
Precondiciones	Esta opción estará disponible si previamente el usuario ha validado sus datos en el sistema, siendo éste de perfil Consultor. Además, deben existir enlaces creados.

Postcondiciones	Enlace eliminado de la base de datos y del tablero.
Escenario básico	- Una vez iniciada la sesión del usuario perfil Consultor, se podrán borrar enlaces. - Para ello, se ha de acceder al panel de creación de Red (Archivo – Crear Red). - Se introduce el ID del enlace a borrar y se pulsa Aceptar.

Tabla 9. Caso de uso Borrar Enlace

3.3. Arquitectura del Sistema desarrollado

La arquitectura del sistema se ha diseñado en función de los objetivos prefijados anteriormente así como por las restricciones de hardware encontradas. No obstante, la solución desarrollada es flexible, de forma que se pueden realizar mejoras y ampliaciones de funcionalidades, tal y como se detallará en el capítulo de líneas futuras.

Básicamente, se precisa que un usuario, mediante un lector RFID y un PC, cree una red y quede posicionado en un escenario (gracias a la información que transmiten las etiquetas RFID), visualizándose dicha red mediante una cuadrícula.

Teniendo en cuenta la aplicación que se quería desarrollar y los dispositivos con los que se contaba, la arquitectura del sistema se muestra en la Figura 9.

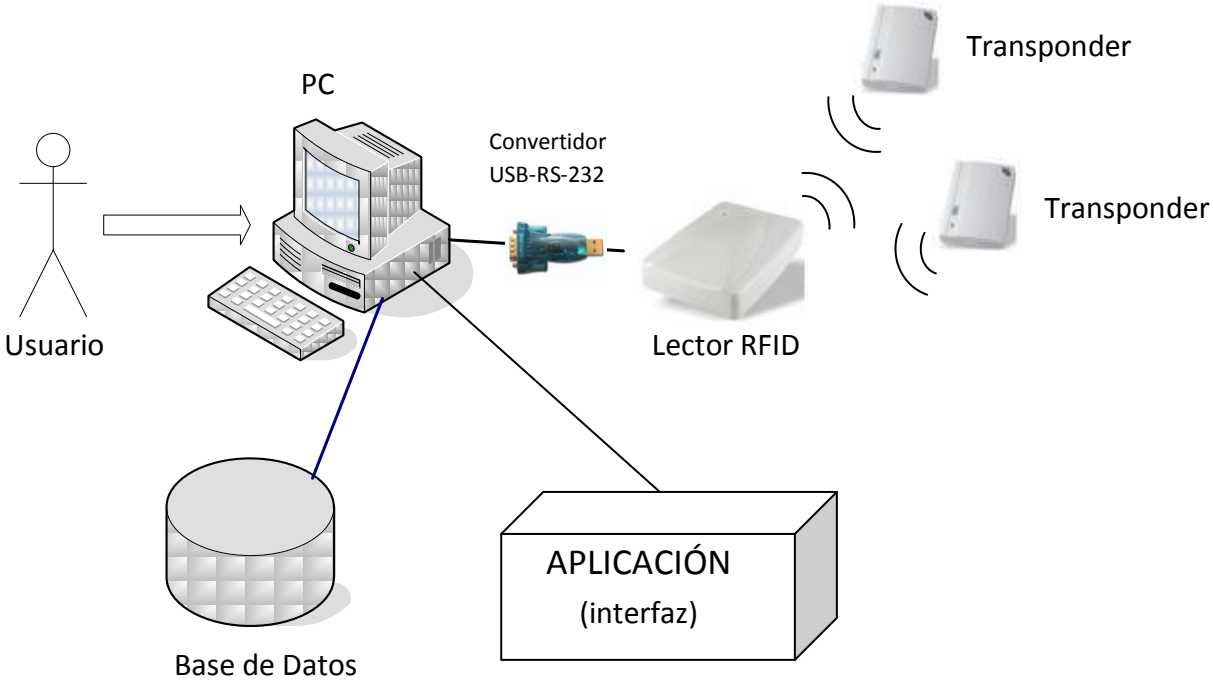


Figura 9. Arquitectura del sistema

Para la realización de este sistema se ha contado con una arquitectura compuesta por dos tags RFID activas de lectura de larga distancia que transmiten en frecuencia microondas, un lector RFID activo para leer los datos que transmiten las etiquetas, que tiene como interfaz de conexión un RS-232, y un PC al que se conecta dicho lector por medio de un hardware convertidor a USB, que virtualiza a puerto COM el mencionado RS-232 de forma que el sistema pueda ser utilizado por cualquier PC. Cabe recordar, que en la actualidad, el uso del puerto RS-232 está quedando prácticamente obsoleto, por este motivo es por lo que se ha decidido usar el conversor.

Se ha optado por utilizar elementos RFID activos, tanto etiquetas como lector, porque garantizan unos resultados y una fiabilidad mayor en cuanto al alcance, y porque eran los elementos con los que contaba el laboratorio que ha facilitado el hardware que se utiliza en este proyecto.

La información que se extrae de la comunicación entre las etiquetas y el lector RFID estará gestionada por uno de los módulos implementados en la aplicación, la cual se explicará en detalle más adelante. En resumen, cuando el lector recibe la instrucción de lectura de datos,

emite una señal al ambiente con el fin de encontrar etiquetas RFID. En caso de existir alguna etiqueta en el rango de influencia (de alcance) del lector, cada una de éstas devolverá su identificador, que es unívoco, y el lector obtendrá, a su vez, la potencia (RSSI) recibida por parte de cada una de ellas. Esta potencia tendrá un valor que oscilará entre 0 y 255, de forma que cuanto más cerca se encuentre la etiqueta RFID del lector, más potencia recibirá éste de la etiqueta.

Tal y como se puede observar en la arquitectura del sistema, en el mismo existe una parte dedicada a la interacción del usuario con el software. Para ello, el usuario utilizará la interfaz que la aplicación le ofrece, de tal forma que se establezca un flujo de información que alimente al sistema, pues el usuario, según el perfil con el que se conecte, participará en el sistema recogiendo, introduciendo y visualizando información gracias a la interfaz.

La aplicación implementada, por su parte, tiene como punto de partida el inicio de sesión del usuario, para posteriormente mostrar las funcionalidades a las que tiene permiso el usuario según el perfil con el que se inicie dicha sesión.

Como ya se indicó anteriormente, el objetivo principal es crear una red posicionando al usuario en un escenario, con la posibilidad de crear enlaces entre los distintos posicionamientos por lo que, para conseguir llegar a ello, previamente se debe realizar un proceso de identificación o señalización del espacio (escenario), de forma que el usuario de perfil Administrador podrá crear puntos que identifiquen una posición en el escenario para que posteriormente, el usuario Consultor sepa dónde está localizado.

Este proceso, como ejemplo en términos prácticos, sería el siguiente:

Se supone que se quiere señalizar un escenario, una sala, en la que posteriormente se posicionará al usuario. En primer lugar, habría que estudiar dónde y a qué distancia se colocarían físicamente las etiquetas RFID en este espacio. En el capítulo de Pruebas se dará una orientación de la eficiencia, en función de la distancia, de estas etiquetas. Una vez colocadas, el usuario Administrador podrá añadir puntos en el espacio, de tal forma que elegirá las coordenadas en las que guardar información. Simultáneamente, y de forma interna, la aplicación almacenará también la potencia extraída de cada unas de las etiquetas. Para intentar que sea una medida fiable y correcta, por cada etiqueta se realizarán tres tomas de “consulta de potencia” prácticamente inmediatas y se calculará la media de éstas, siendo el resultado la potencia final obtenida en ese punto (x,y).

Una vez "señalizado" el espacio en el que se está trabajando, es decir, que cada punto esté señalizado con su respectiva potencia, se estaría en disposición de localizar a cualquier usuario que entrara en dicho espacio. Para ello, se crea la figura del usuario Consultor. Este perfil tiene la posibilidad de crear una red en la que visualizar, por medio de las funcionalidades existentes, en qué punto del espacio está situado el usuario en un momento concreto, y los enlaces entre varios de esos puntos.
Para este caso, a los puntos se les llamará nodos, de forma que un usuario, cuando quiera ser posicionado, creará un Nodo que estará en un punto del espacio.
De esta forma, el usuario, según se vaya moviendo por el espacio, podrá crear nodos en algunos, o en todos, aquellos puntos en los que se haya situado gracias a las potencias que reciba en ese momento de cada etiqueta.
Si, por ejemplo, en el espacio señalizado la posición (5,5) tiene un valor de potencia media de 150 de la etiqueta1 y un valor de 160 de la etiqueta2, cada vez que un usuario cree un nodo y reciba internamente esas potencias, se podrá asegurar que la posición en la que está es la citada (5,5).
Además, una vez creados varios nodos, se podrán unir por medio de enlaces.

Por último, simultáneamente a este proceso y como aspecto imprescindible (ya que el usuario interactúa introduciendo y recibiendo información generada en base a otros datos), toda la información generada se almacena en una Base de Datos espacial Oracle, es decir, todas las operaciones que conlleven inserciones, borrados o consultas se realizarán sobre las correspondientes tablas de una base de datos. Este tipo de almacenaje es muy importante, pues se precisan operaciones de autenticación de usuarios o creación de redes por medio de geometrías, y de esta forma, el usuario podrá recuperar la información generada en otras sesiones (conexiones al sistema) realizando consultas a la base de datos, es decir, accediendo a los datos que ésta tiene almacenada.

Una vez que se ha descrito el sistema global, se pasa a diseñar los componentes del mismo: la base de datos; el diagrama de clases necesario para la aplicación encargada de recibir, mostrar y gestionar información; y la comunicación RFID que tendrá lugar.

3.4. Diseño de la Base de Datos

En este apartado, una vez identificada toda la información susceptible de ser almacenada en la Base de Datos, se creará el modelo entidad-relación (que se explicará en detalle), y se elaborará el esquema relacional.

3.4.1. Modelo Entidad-Relación

En este apartado se detalla la elaboración del modelo entidad-relación. En primer lugar se analizan las entidades participantes en el esquema, así como los atributos que tienen para, posteriormente, analizar las interrelaciones existentes, dando como resultado el esquema entidad-relación completo.

Entidades y atributos

Una entidad es aquel objeto sobre el que se quiere guardar información en la base de datos, mientras que los atributos se utilizan como propiedades descriptivas de las mismas. Teniendo esto en cuenta se extraen las siguientes entidades y atributos:

1. **Usuario**: esta entidad representa a las personas que se conectarán a la aplicación, de las cuales será necesario almacenar información, especialmente los datos relativos al inicio de sesión en el sistema. La entidad Usuario y sus atributos se muestran en la Figura 10.

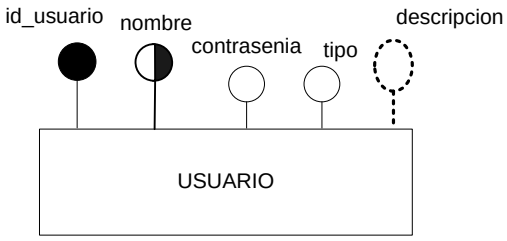


Figura 10. Entidad Usuario

   - *id_usuario*: es el identificador del usuario, la clave primaria del mismo, por lo que es obligatorio. Está formado por un número entero que se incrementa de uno en uno, como un contador, por cada inserción que se realiza.

   - *nombre*: almacena el nombre de usuario. Es un atributo candidato a ser clave primaria puesto que no habrá dos usuarios con el mismo nombre, de ahí, que sea representado como clave alternativa.

- *contrasenia*: almacena la contraseña del usuario. Está formado por caracteres alfanuméricos, al igual que el nombre. Junto a éste forman los datos de acceso a la aplicación por parte del usuario.

- *tipo*: identifica el tipo de usuario que se conecta a la aplicación. Para este proyecto se han pensado dos perfiles: Administrador y Consultor, que naturalmente pueden verse ampliados en caso de necesidad futura. El perfil Administrador señala los puntos del escenario y el usuario con perfil Consultor crea la red con los nodos y enlaces correspondientes.

- *descripcion*: es un atributo no obligatorio en el que se almacena una breve descripción del usuario en caso necesario. Se ha representado con líneas discontinuas en su totalidad.

2. **Punto**: esta entidad representa cada punto existente dentro del escenario. Es una entidad muy importante puesto que para poder posicionar a un usuario a la hora de crear la red, previamente, el escenario debe haberse marcado con puntos y con sus correspondientes potencias respecto a los tags existentes. La entidad Punto se muestra en la Figura 11.

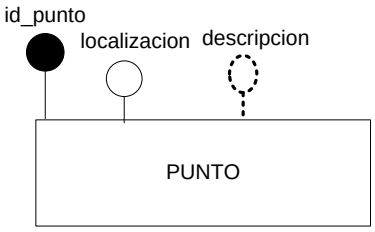


Figura 11. Entidad Punto

- *id_punto*: es el atributo identificador del punto, la clave primaria del mismo, por lo que es obligatorio. Está formado por números enteros que se incrementan de uno en uno, como un contador, por cada inserción que se realiza.

- *localización*: es el atributo que indica la posición del punto, es decir, su posición x,y en los ejes de coordenadas, teniendo en cuenta que cada casilla del escenario se relaciona con una posición en el eje. Tiene la particularidad de ser un atributo geo-espacial. De tal forma que para crear un punto, el usuario Administrador debe facilitar las coordenadas x e y del mismo, que se

guardarán en el atributo localización. Por tanto, es un atributo obligatorio. Además, es un atributo único pues no sería coherente que dos puntos se encuentren en la misma posición (coordenadas), a la par que complicaría la creación de la red.

- *descripcion*: es un atributo no obligatorio en el que se almacena una breve descripción del punto en caso necesario.

3. **Tag**: es también una entidad imprescindible del sistema pues se precisa su información para conocer la potencia de cada punto en la red. La entidad Tag se muestra en la Figura 12.

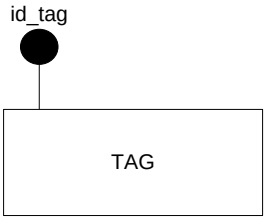


Figura 12. Entidad Tag

- *id_tag*: es el atributo identificador del tag, la clave primaria del mismo, por lo que es obligatorio. A diferencia de los anteriores atributos identificadores, se conoce que los tags tienen (de serie) un identificador unívoco de 16 caracteres. Este identificador se extrae de la comunicación que se establece entre el lector y el tag.

4. **Nodo**: representa las posiciones por las que el usuario se posiciona dentro de la red. Estos nodos se posicionarán en puntos de la red en función de la potencia obtenida. Esta entidad se representa en la Figura 13.

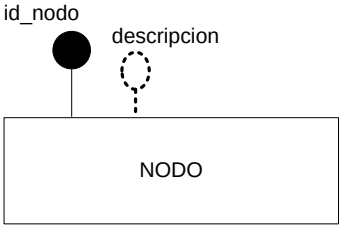


Figura 13. Entidad Nodo

- *id_nodo*: es el atributo identificador del nodo, la clave primaria del mismo, por lo que es obligatorio. Está formado por números enteros que se incrementan de uno en uno, como un contador, por cada inserción que se realiza.

- *descripcion*: es un atributo no obligatorio en el que se almacenan una breve descripción del nodo en caso necesario.

5. **Enlace**: representa la unión de dos nodos en la red. Como se verá más adelante, precisará de ellos para tener sentido por lo que será una entidad débil. Esta entidad se representa en la Figura 14.

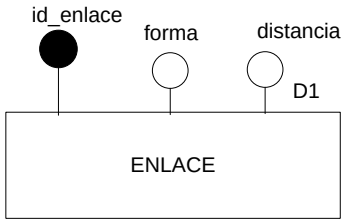


Figura 14. Entidad Enlace

- *id_enlace*: es el atributo identificador del enlace, la clave primaria del mismo, por lo que es obligatorio. Está formado por números enteros que se incrementan de uno en uno, como un contador, por cada inserción que se realiza.

- *forma*: es el atributo que indica la forma del enlace con la particularidad de ser un atributo geo-espacial. Por tanto, en él se definen los datos correspondientes para formar la línea que une los dos nodos que conforman el enlace. Es un atributo obligatorio pues esta información es imprescindible para representar el enlace y además es único, pues no existirán dos enlaces exactamente iguales.

- *distancia*: indica el coste del enlace, es decir, la distancia existente entre los dos nodos que lo forman. Este atributo es derivado, y la forma de darle valor es mediante el cálculo de la distancia euclídea, que se obtiene mediante la fórmula que se muestra en la Figura 15.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Figura 15. Fórmula matemática de la distancia euclídea

Una vez analizadas las entidades y sus atributos, se deben identificar las relaciones existentes entre ellas, por lo que a continuación, junto a las ilustraciones correspondientes, se detallan dichas relaciones:

- Relación entre las entidades Usuario y Nodo

  Como se puede observar en la imagen, cada usuario se sitúa en un nodo en un momento dado, de ahí que la cardinalidad máxima y mínima sea 1.
  En cambio, un nodo puede representar de 0 a N usuarios, por lo que la cardinalidad mínima es 0 y la cardinalidad máxima es N. De esta forma, el tipo de correspondencia será 1:N, la relación se representa en la Figura 16.

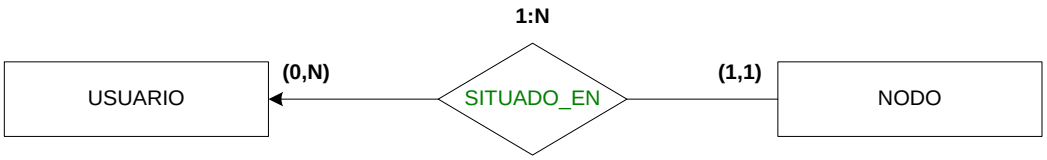


Figura 16. Interrelación entre las entidades Usuario y Nodo

- Relación entre las entidades Punto y Tag

  En esta relación, cada Punto consulta todas las tags existentes en el sistema, de forma que la cardinalidad mínima es 1 (teniendo en cuenta que habrá como mínimo una etiqueta RFID) y la cardinalidad máxima N etiquetas. Por otro lado, de forma análoga, cada Tag puede leerse en cada punto del sistema, por lo que la cardinalidad mínima es 1 (como mínimo habrá un punto que consulte el tag) y la cardinalidad máxima es N. El tipo de correspondencia, por tanto, es N:M. La interrelación, además, tiene un atributo: *potencia_media*, que indica la potencia media obtenida de un tag en un punto determinado. La relación se representa en la Figura 17.

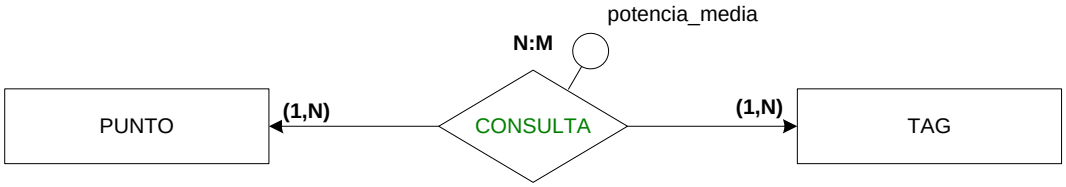


Figura 17. Interrelación entre las entidades Punto y Tag

- Relación entre las entidades Nodo y Enlace

  En esta relación se observa una dependencia en existencia, ya que el enlace (entidad débil) depende de la existencia de un nodo (entidad fuerte) pues un enlace está formado por dos nodos que, naturalmente, deben existir.

  Para analizar las cardinalidades, se debe tener en cuenta que un enlace está formado por dos y sólo dos nodos (cardinalidades mínima y máxima) y un nodo puede no participar en ningún enlace o hacerlo en N enlaces. En este caso, se ha de tener en cuenta que el usuario no tiene por qué señalar todos los posibles enlaces de la red, de forma que puede darse el caso que un nodo se quede sin participar en ningún enlace de la misma, ya que esta aplicación se centra en la construcción de la red. Como se puede observar en la Figura 18, el estudio de estas cardinalidades deriva en un tipo de correspondencia 2:N.

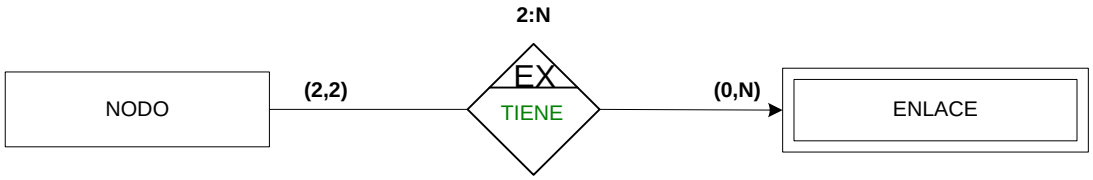


Figura 18. Interrelación entre las entidades Nodo y Enlace

- Relación entre las entidades Nodo y Punto

  En esta relación se observa una dependencia en existencia, ya que el nodo (entidad débil) depende de la existencia de un punto en el escenario (entidad fuerte), pues un nodo se sitúa en la red sobre un punto.

  Para analizar las cardinalidades, se debe tener en cuenta que un nodo está sobre uno y sólo un punto (cardinalidades mínima y máxima igual a 1). Además, en cuanto a

cuantos nodos puede tener un punto, como ya se ha indicado, las redes se crearán en un determinado espacio o escenario sobre el que se realizan dos fases:

1) Señalización del escenario: Ubicación de puntos en el espacio que se debe tener en cuenta. Es decir, dividir el espacio como una cuadricula y a cada celda de la misma asociarle un punto (coordenadas más potencia). Esta señalización cubre todo el espacio y es independiente de la aplicación posterior que se le vaya a dar, de manera que es independiente del escenario. Un ejemplo de esta relación se puede ver en la Figura 19.

2) Creación de redes: Se crea una red por cada escenario que se quiera tener en cuenta basándose en la señalización anteriormente realizada. Esto se puede ver en la Figura 19 en las capas con nombres "Escenario 1" y "Escenario 2".

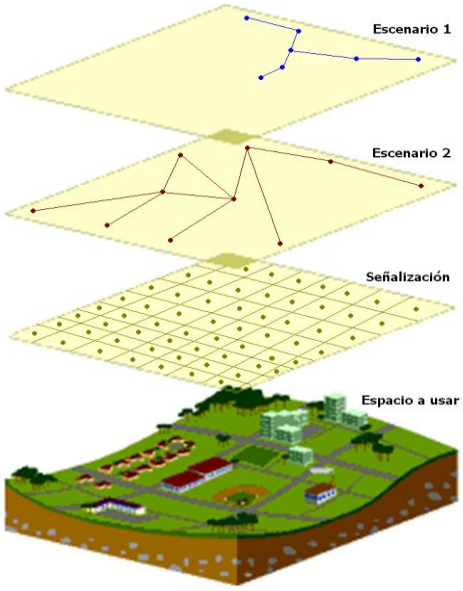


Figura 19. Señalización y creación de escenarios

Por todo esto, se tendrá que un punto puede no tener ningún nodo en su posición o bien, tener N nodos en función del posicionamiento tomado por el usuario en el escenario (cardinalidad mínima 0 y cardinalidad máxima N).

Como se puede observar en la Figura 20, el estudio de estas cardinalidades deriva en un tipo de correspondencia 1:N.

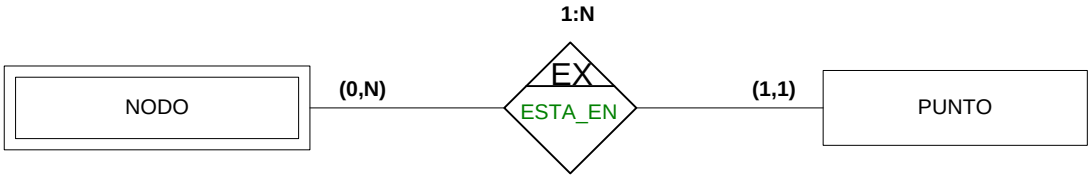


Figura 20. Interrelación entre las entidades Nodo y Punto

En base a todo el análisis realizado, se está en disposición de mostrar el esquema entidad-relación completo, que da cabida a toda la información que se recogerá en el sistema. Ésta se muestra en la Figura 21.

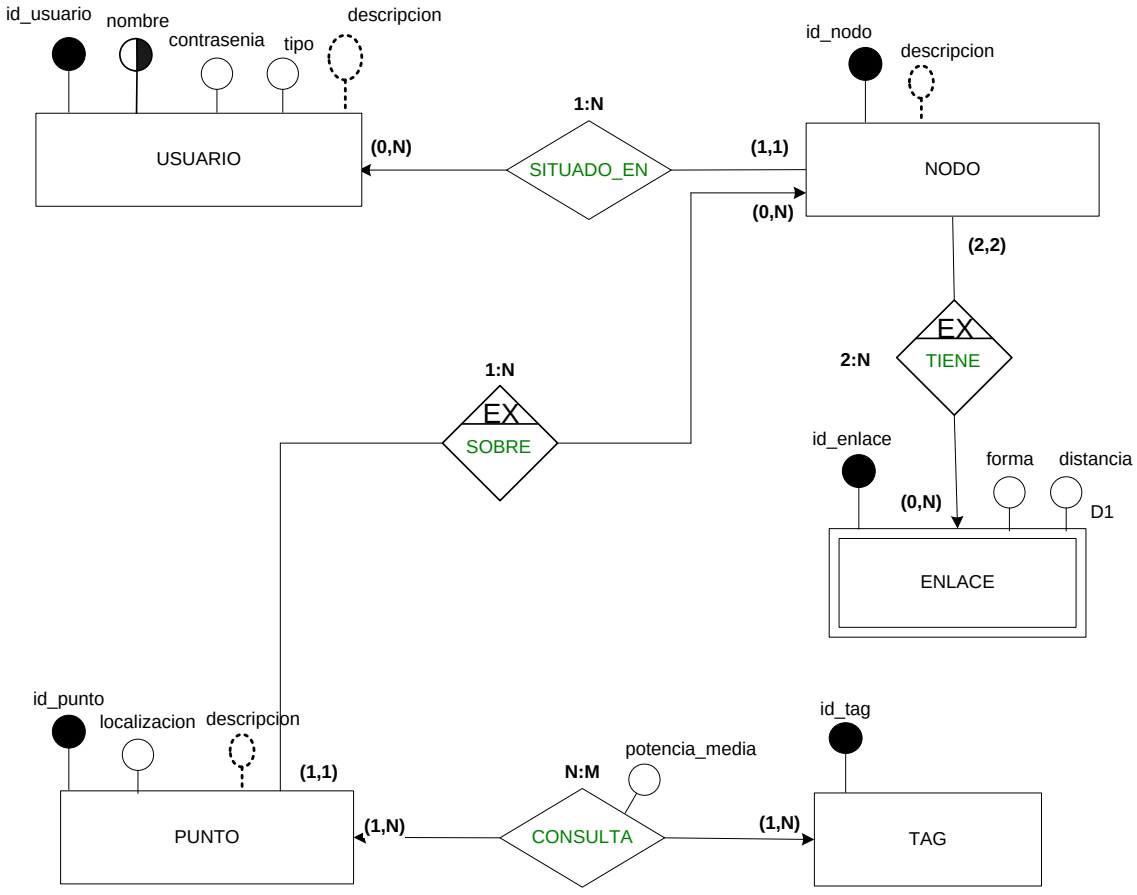


Figura 21. Esquema entidad-relación

Supuestos semánticos no recogidos

En el esquema entidad-relación no se han recogido los siguientes aspectos:

- El atributo derivado *distancia* de la entidad **Enlace** se calcula a partir de las posiciones de los nodos que forman cada enlace y que pertenecen a la entidad **Nodo**.

- El atributo *tipo* de la entidad **Usuario** sólo puede tomar dos valores: Administrador o Consultor.

- Las claves primarias que identifican cada entidad tienen valores consecutivos, incrementan un número por cada instancia.

3.4.2. Esquema Relacional

A continuación se expone el modelo relacional del sistema. Para esto, se realizará una transformación del modelo Entidad-Relación detallado anteriormente, y se tendrán en cuenta las restricciones inherentes al mismo [40] [41]. En dicha transformación se ha de tener en cuenta que cada entidad se transforma en una relación, las interrelaciones con tipo de correspondencia 1:N ó 0:N dan lugar a una propagación de clave o una relación (normalmente el primero de los casos), y que las interrelaciones N:M dan lugar a una nueva relación.

- Siguiendo las pautas anteriores, la entidad **Usuario** es una nueva relación (tabla) del esquema. Su clave primaria es *id_usuario* puesto que identifica unívocamente a cada uno de ellos. El resto de atributos deben ser incluidos en la relación. La interrelación **Situa_en** es 1:N por lo que da lugar a una propagación de clave, de forma que en esta relación USUARIO se añade un nuevo atributo *nodo*, que tiene la función de clave ajena de la relación NODO. Las operaciones de borrado y modificación serán en Cascada (DC,UC) ya que si se borra un nodo cuyo identificador está relacionado con un usuario, éste debe ser borrado debido a que a partir de ese momento carece de sentido su existencia. La Figura 22 muestra esta transformación.

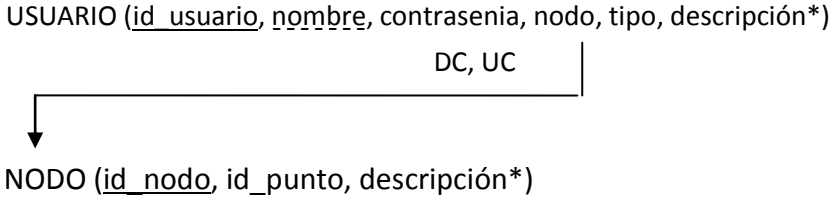


Figura 22. Transformación de la entidad Usuario

- La entidad **Punto** es una nueva relación (tabla) del esquema. Su clave primaria es *id_punto* puesto que identifica unívocamente a cada uno de ellos. El resto de atributos

deben ser incluidos en la relación. La interrelación **Esta_en** es 1:N por lo que da lugar a una propagación de clave, de forma que en la relación NODO se añade un nuevo atributo id_*punto*, que tiene la función de clave ajena de la relación PUNTO. Esta propagación se ha llevado a cabo de esta manera pues un nodo estará sobre un punto ya creado previamente. Las operaciones de borrado y modificación serán en Cascada (DC, UC) ya que si se borra un punto cuyo identificador está relacionado con un nodo, éste debe ser borrado debido a que a partir de ese momento carece de sentido su existencia. Esta transformación se muestra en la Figura 23.

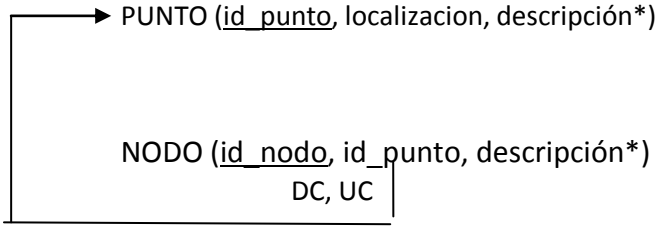


Figura 23. Transformación de la entidad Nodo

- La entidad **Punto**, como ya se ha indicado, y la entidad **Tag** (con clave primaria *id_tag*) se convierten en nuevas relaciones. En el esquema E/R se puede observar que la interrelación **Consulta**, en la que participan ambas entidades, es N:M por lo que se debe crear una nueva tabla, cuyo nombre será PUNTO_TAG. Esta nueva relación estará formada por la clave principal de las relaciones que participan en ella, es decir, la clave primaria de PUNTO (*id_punto*) y la de TAG (*id_tag*) que además, conjuntamente, formarán la clave primaria de esta nueva tabla, con el objetivo de evitar la duplicidad de tuplas. Además, esta relación cuenta con un atributo, *potencia_media*, que almacena la media de las tres potencias obtenidas en las tres tomas realizadas. Las opciones de borrado y modificación serán en cascada ya que si en un caso concreto no hay punto ni tag, no debe haber una tupla en esta tabla intermedia. La Figura 24 muestra esta información.

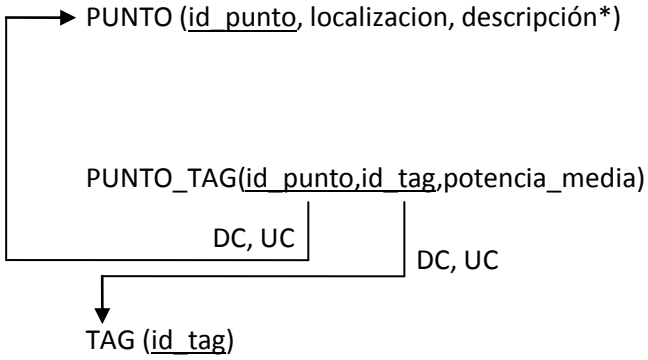


Figura 24. Tranformación de la interrelación Punto_Tag

- La entidad **Enlace** se convierte en una nueva tabla cuya clave primaria es *id_enlace*. Como se puede observar en el esquema E/R existe una interrelación de dependencia en existencia, llamada **Tiene**, entre la entidad **Nodo** y la entidad **Enlace**, con cardinalidad 2:N, por lo que se precisa una propagación de claves. Para este caso concreto, se propaga la clave primaria de la relación NODO (*id_nodo*) como clave ajena a la relación ENLACE. La particularidad existente es que esta clave se encontrará dos veces (*id_nodo_orig*, *id_nodo_dest*) en esta relación, pues un enlace está formado por dos nodos. Además, como por cada par de nodos no puede haber varios enlaces, la unión de dichos atributos se toma como clave alternativa.

  Las opciones de borrado y modificación son en cascada ya que si los nodos que forman el enlace no existen, no es preciso almacenarlo, pues deja de tener sentido su existencia. Esta información se refleja en la Figura 25.

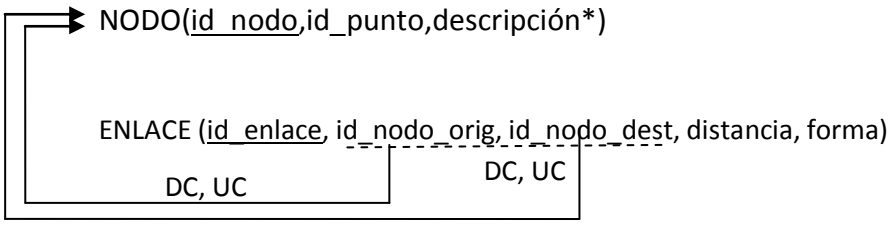


Figura 25. Transformación de la entidad Enlace

Finalmente, agrupando todos los pasos que se han ido tomando, se obtiene el esquema relacional completo, tal y como se muestra en la Figura 26.

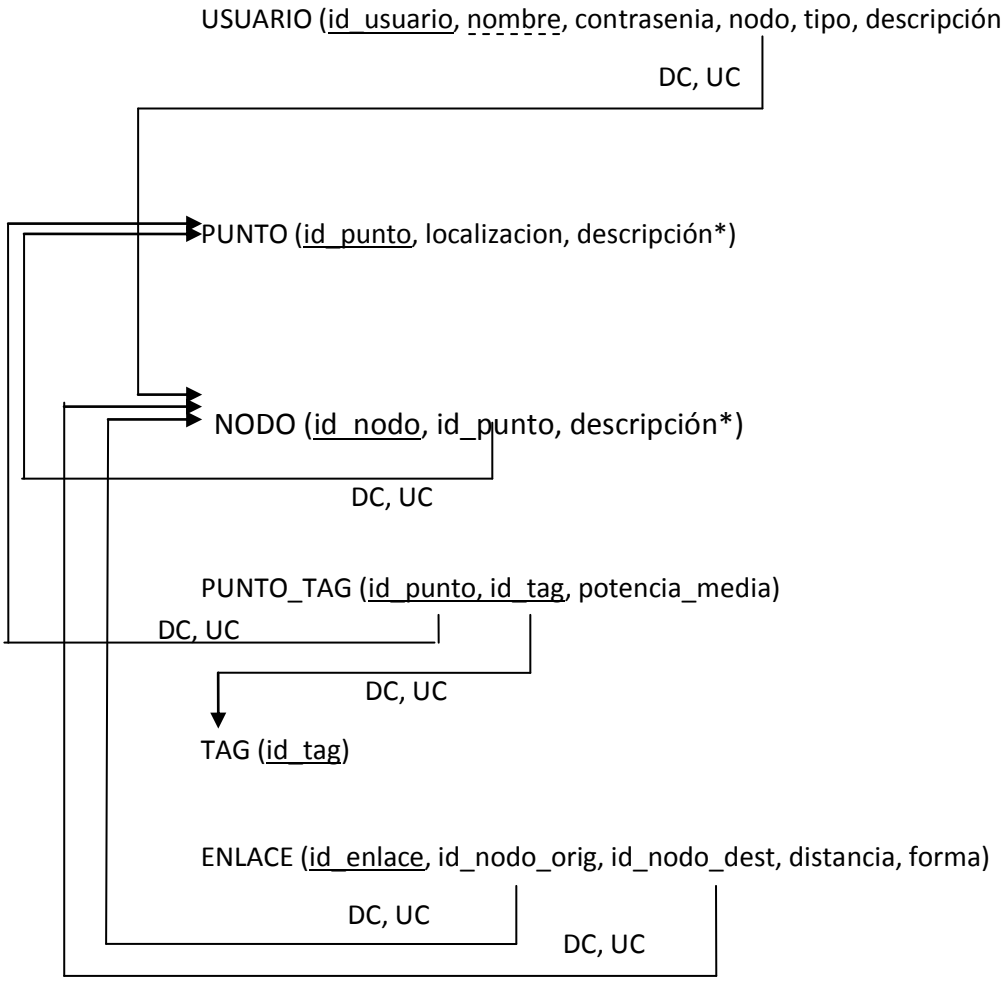


Figura 26. Esquema relacional

Supuestos semánticos no reflejados

- Las cardinalidades mínimas y máximas en cada relación.
- Atributos derivados, como *distancia*, de la tabla ENLACE que es un atributo derivado.
- El dominio del atributo *Tipo* de USUARIO. Este atributo puede tener dos valores: Administrador o Consultor.

3.5. Diagrama de clases

El diagrama de clases que se presenta es el resultado del diseño realizado, y es el mostrado en la Figura 27. En él se pueden ver las clases que son necesarias para el correcto funcionamiento del sistema.

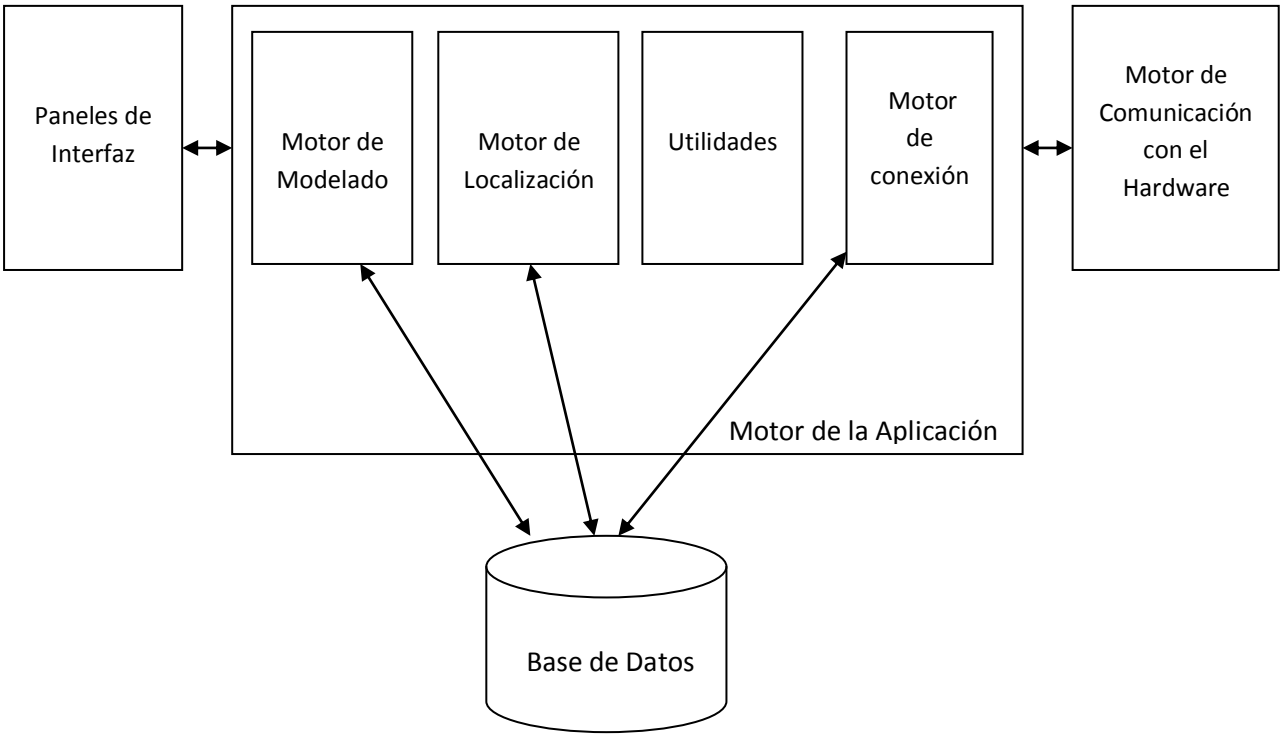


Figura 27. Diagrama de clases

El motor de la aplicación está formado por las clases que realizan las operaciones de modelado y localización (motor de modelado y motor de localización), con la ayuda de las clases de utilidades y de las conexiones con la base de datos, que es imprescindible para almacenar y obtener datos cuando se precise. A su vez, el motor de conexión realiza la funcionalidad de conexión y desconexión del usuario con la aplicación.

El motor de comunicación con el hardware permite establecer la comunicación con el lector, de forma que se puedan interpretar correctamente las tramas que se envían las etiquetas y el propio lector.

A su vez, para presentar la información por pantalla se utilizan las distintas funcionalidades incluidas en los paneles de interfaz.

3.6. Diseño Comunicación RFID

En este apartado se detallan las especificaciones y usos de las tags y del lector RFID.

3.6.1. Dispositivos RFID

El sistema RFID utilizado para el desarrollo del proyecto consta de dos tags o etiquetas RFID (transpondedores) y un lector de RFID para recibir datos de las tarjetas.

En este proyecto no se va a incluir un estudio comparativo de los dispositivos usados con otros de modelos similares debido a que eran éstos de los que se disponía para la realización del trabajo. No obstante, se detallan sus características y sus principales ventajas e inconvenientes respecto a otros modelos similares.

Etiquetas RFID

En cuanto a las etiquetas RFID, se ha contado con etiquetas de tipo activas, notablemente más eficientes y completas que las etiquetas de tipo pasivo. Los tags activos [42] poseen una fuente de energía que permite que éstos puedan leerse a mayor distancia respecto a los tags pasivos. Además, responden a señales de menor nivel e incluso, debido a la mayor cantidad de memoria interna que poseen, pueden tener ciertas funcionalidades que otros tags no tienen, tales como sensores medioambientales.

En concreto, el modelo de tag activo utilizado se muestra en la Figura 28 y sus especificaciones son las siguientes [43]:

- Tag RFID Activo SYTAG245-2K

Figura 28. Tag RFID Activo SYTAG245-2K

Descripción:

Tag RFID de lectura desde larga distancia transmitiendo en frecuencias de microondas (2,45 GHz). Los tags serán identificados, localizados y trazados de manera cómoda, segura y fiable en cualquier sitio de su instalación.

Características radio

Frecuencia de comunicación: 2,45 GHz
Rango de frecuencia: 2,40 ~ 2,48 GHz
Canal: 255
Dirección: 65536 direcciones
Wake on radio: ON / OFF
RSSI: 0-255
ID: 64 bits
Programación: configurable a partir de comandos
Led: acción o estatus
Conmutación: configurada como tag activo o tag ON / OFF
Memoria: 4 kbytes ~ 32 kbytes (opcional)

Alimentación

Batería: 3 VDC CR2032
Consumo en reposo: 3 uA @ 3 VDC
Consumo en funcionamiento: 24 mA @ 3 VDC

Entorno

Temperatura de funcionamiento: -10 ºC a 55 ºC
Temperatura de almacenaje: -20 ºC a 65 ºC
Humedad relativa de funcionamiento y almacenaje: 5% al 95%

Dimensiones

Dimensiones: 42 x 30 x 10 mm

Aspectos a destacar

Ventajas

Entre las ventajas de este modelo de tag destacan: una aceptable distancia de lectura, el bajo consumo de su batería, resistencia al agua, así como un coste ajustado respecto a otros modelos.

Inconvenientes

Entre los inconvenientes de este modelo de tags destaca su diseño, ya que son etiquetas pequeñas pero que difícilmente pueden ser utilizadas en algunos productos como, por ejemplo, en productos de un supermercado o en la cadena de suministro. Además, se pueden producir problemas de detección de las mismas si en el escenario se encuentran obstáculos, aunque sean pequeños, aspecto que no sucede con otros modelos de etiquetas.

Lector RFID

En cuanto al lector RFID, se ha utilizado un dispositivo que es capaz de leer información de tags localizados a unos metros de distancia. En concreto, el modelo utilizado se muestra en la Figura 29 y sus especificaciones son las siguientes [44]:

- Lector RFID Activo SYRD245-2

Figura 29. Lector RFID Activo SYRD 245-2

Descripción:

Lector de tags RFID Activo, permite la lectura de datos de tags desde larga distancia transmitiendo en frecuencias de microondas (2,45 GHz).

Características radio

Frecuencia de comunicación: 2,45 GHz

Rango de frecuencia: 2,40 ~ 2,48 GHz

Canal: 255

Dirección: 65536 direcciones

RSSI: 0-255 (permite determinar cuál es el lector más cercano al tag en caso que las áreas de lectura de varios dispositivos intersecciones)

LQI: 0-255

Programación: configurable a partir de comandos

Led: acción o estatus

Rango de lectura: hasta 5 metros

Interface

RS-232: RS, TX (SYRD245-2S)

Baud rate: 2.400 bps ~ 115.200 bps

Alimentación

Alimentación de entrada: 3,3 VDC ~ 12 VDC (RS-232);

Consumo de corriente: 100 mA @ 5 VDC

Entorno

Temperatura de funcionamiento: -20 ºC a 65 ºC

Temperatura de almacenaje: -30 ºC a 85 ºC

Humedad relativa de funcionamiento y almacenaje: 5% al 95%

Dimensiones

Dimensiones: 92 x 66 x 20 mm

Aspectos a destacar

Ventajas

Entre las ventajas de este modelo de lector RFID destacan: una larga distancia de lectura, capacidad de lectura multi-tag que gestiona las colisiones que se puedan producir entre ellas, seguridad y fiabilidad, antena integrada, así como un coste ajustado respecto a otros modelos similares.

Como se puede observar en el apartado Interface de las especificaciones de este dispositivo, la conexión a un PC se realiza por medio del puerto RS-232, que se detalla en el siguiente apartado.

Inconvenientes

El modelo de lector RFID utilizado para este proyecto presenta un inconveniente en cuanto al Interface, es decir, en cuanto a la conexión del mismo con el PC. La conexión se realiza mediante el puerto RS-232, que con el paso de los años prácticamente se ha convertido en obsoleto.
Esto limita mucho el rango de aplicación del lector, pues en los equipos actuales no es común encontrar dicho puerto.

Existen cables conversores de puerto RS-232 a USB que, no obstante, hacen que el sistema sea más complejo a la hora de implementar un módulo que recoja los datos que el lector ofrece.

3.6.2. Puerto RS-232. Convertidor a USB

Con el fin de trabajar con la información que se extrae de la comunicación establecida entre las etiquetas y el lector RFID (y para que éste funcione), el puerto RS-232 será el intermediario entre el software implementado y el lector RFID. Como se indicaba anteriormente, el modelo de lector RFID utilizado tiene como interfaz de conexión un puerto RS-232. Para utilizar un PC portátil moderno que no tuviera esta opción de conexión y, con la motivación de ahondar en la investigación de este hardware, se ha decidido probar un convertidor a USB, similar al que se presenta en la Figura 30, que "convirtiera" el puerto de conexión. Tras realizar la instalación de los drivers correspondientes, y tras las pruebas

pertinentes, se comprobó que el funcionamiento utilizando este convertidor era exitoso, y por ello se integró dentro del sistema creado.

Figura 30. Convertidor de puerto RS-232 a USB

Para obtener la información que el lector y las tarjetas se envían, y aunque en el capítulo de Implementación se explicará en detalle, ha sido preciso utilizar librerias JAVA con funciones que permitiesen leer datos e información del puerto COM (puerto utilizado). El convertidor virtualiza internamente, gracias a sus drivers, el puerto USB a puerto COM, de forma que dicho puerto USB en el que se conecta el convertidor pasa a ser un puerto COM de número aleatorio. En el caso del PC utilizado para realizar las pruebas del sistema, el puerto ha sido COM4.

El conector COM con el que se ha contado ha sido de 9 pines DB-9 [45] tal y como muestra la Figura 31.

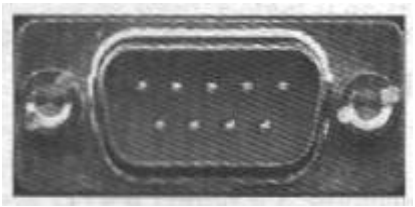


Figura 31. Conector COM de 9 pines

Cada pin puede ser de entrada o de salida y cada uno de ellos tiene una función diferente, tal y como se indica en la Figura 32 y en la Tabla 10.

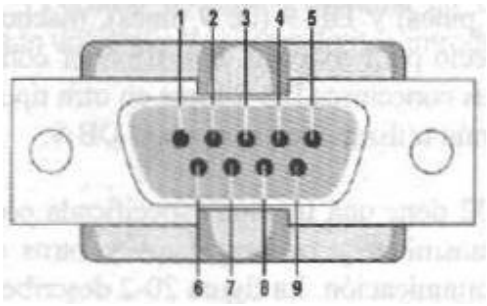


Figura 32. Pines de conector COM

PIN	FUNCIÓN
1	Detección de Portadora (Data Carrier Detect - DCD)
2	Recibir Datos (Received Data - RxD)
3	Transmitir Datos (Transmitted Data - TxD)
4	Terminal de datos listo (Data Terminal Ready - DTR)
5	Tierra de Señal (Signal Ground - SG)
6	Equipos de Datos listo (Data Set Ready - DSR)
7	Solicitud de envío (Request to Send - RTS)
8	Libre para envío (Clear to Send - CTS)
9	Indicador de llamada (Ring Indicator - RI)

Tabla 10. Funciones de los pines de un conector COM

Las señales TxD, DTR y RTS son de salida, mientras que RxD, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es SG (Señal de Tierra). Además, existen otras señales como RI (Indicador de Llamada), y otras poco comunes

El puerto de serie RS-232 puede transmitir los datos en grupos de 5, 6, 7 u 8 bits, a unas velocidades determinadas (normalmente, 9600 bits por segundo, aunque para este sistema se han utilizado 115200 bits por segundo). Después de la transmisión de los datos, le sigue un bit opcional de paridad (indica si el número de bits transmitidos es par o impar, para detectar fallos), y después 1 o 2 bits de Stop.

4. Implementación

Una vez analizados los objetivos a llevar a cabo, las herramientas que se van a utilizar, y el análisis y el diseño del sistema, se precisa implementarlo. Es en este capítulo en el que se explica dicha implementación, teniendo en cuenta que las dos partes fundamentales en las que se apoya el sistema son una base de datos espacial Oracle en la que se almacena la información, y una aplicación con interfaz visual programada en el lenguaje orientado a objetos JAVA en el que se incluye, además, todo el tratamiento de la comunicación entre lector y tarjetas RFID.

4.1. Implementación de la Base de Datos

En el siguiente apartado se explican las decisiones tomadas respecto a la implementación del esquema relacional en el SGBD Oracle.

Creación de tablas

En líneas generales, las tablas contienen atributos con tipos similares, en los que destacan los atributos numéricos, los de cadenas de caracteres, y los geométricos (SDO_GEOMETRY). Para comprender y visualizar la implementación de este tipo de atributos, así como la creación de una tabla en general, se muestra el Ejemplo 10 con la creación de la tabla PUNTO, una de las más importantes del sistema, y que además es bastante ilustrativa y completa.

```
CREATE TABLE PUNTO (
   Id_punto NUMBER(4) NOT NULL,
   Localizacion MDSYS.SDO_GEOMETRY NOT NULL,
   Descripcion VARCHAR2(40),
  CONSTRAINT pk_punto PRIMARY KEY (Id_punto)
);
```

Ejemplo 10. Creación de la tabla Punto

Como se puede observar en dicho Ejemplo 10, la creación de tablas se declara con la cláusula CREATE TABLE y a continuación el nombre de la misma. El identificador de la tabla es numérico, implementado por tanto como tipo NUMBER, y en este caso con posibilidad de tener 4 cifras, es decir, el valor del identificador puede valer, como máximo, 9999. Todos los identificadores de cada tabla serán numéricos salvo el identificador del tag, que en principio está compuesto por 16 dígitos. Dicho atributo se

ha decidido declararlo como VARCHAR2 con el objetivo de ser menos restrictivo, y permitir de este modo la inclusión de caracteres si se precisara en un futuro.

Por otro lado cabe destacar el atributo *Localizacion*, de tipo SDO_GEOMETRY. Este atributo es de tipo espacial y logra, por tanto, almacenar espacialmente el punto. Más adelante se analizarán las características de este tipo de atributos.

Los atributos compuestos por caracteres se declaran como VARCHAR2. En este caso, *Descripcion* puede tener como máximo una longitud de 40 caracteres.

Por último, las claves primarias se declaran con la cláusula CONSTRAINT nombre PRIMARY KEY (nombre del atributo).

La tabla **Punto** no contiene claves ajenas, pero la sentencia utilizada para declararlas se muestra en el Ejemplo 11.

```
CONSTRAINT fk_punto FOREIGN KEY (Id_punto) REFERENCES
PUNTO ON DELETE CASCADE
```

Ejemplo 11. Clave ajena de la tabla Punto

Este ejemplo se encuentra en la creación de la tabla **Nodo**, que como se ha indicado en el esquema relacional, tiene un atributo *id_Punto* como clave ajena que referencia a la tabla *Punto* tal y como indica la cláusula REFERENCES. Cabe destacar la declaración de borrado en cascada (ON DELETE CASCADE).

Inserción de registros

Una vez creadas las tablas, las distintas funcionalidades de la aplicación procederán a insertar registros en las mismas, con el objetivo de "alimentar" el sistema. No obstante, previamente, sí deben ser insertados los datos de partida como las tags que participan en el sistema así como los usuarios que accederán a la aplicación.

Además deben ser cargados los disparadores creados. Cabe destacar, que para incrementar automáticamente el valor identificativo de las tablas que los contengan se ha optado por la creación de disparadores, de tal forma que se tiene un disparador para cada una de las tablas con identificadores (tabla Usuario, Punto, Nodo y Enlace).

Un ejemplo de disparador (en este caso para el identificador del punto) se refleja en el Ejemplo 12.

```
CREATE OR REPLACE TRIGGER ID_PUNTO
BEFORE INSERT ON PUNTO
REFERENCING NEW AS New
FOR EACH ROW
DECLARE
contador NUMBER;
BEGIN
contador:=0;
SELECT NVL(MAX(Id_punto),0) +1 INTO contador
FROM Punto;
:NEW.Id_punto:=contador;
END;
/
```

Ejemplo 12. Disparados del identificador de Punto

Este disparador (trigger) indica que antes de insertar un tupla en la tabla, y para cada fila, se debe aumentar en uno el identificador en función del último de registro. Para ello, se crea un contador, consulta el valor máximo del identificador en cuestión, que al fin y al cabo será el último, y le suma uno. De esta forma, la nueva tupla contendrá el valor del identificador actualizado.

Otra posible opción para el autoincremento de los valores de los identificadores sería la creación de secuencias. Sin embargo, se ha optado por la creación de disparadores, a priori más complejos de gestionar, por la mayor familiarización existente con estos últimos en lugar de secuencias.

No obstante, en el Ejemplo 13 se presenta un ejemplo de creación de secuencia para autoincrementar un identificador.

```
CREATE SEQUENCE incremento_id_punto
INCREMENT BY 1
START WITH 1
```

Ejemplo 13. Secuencia para incrementar el identificador de Punto

Como se observará a continuación, en las inserciones cuyo identificador se actualice con disparadores, no será preciso indicar ningún valor en dicho campo. Por tanto, en estas inserciones ese campo quedará representado con ‘ ’. En el caso de haber usado

secuencias, este campo debería llevar el nombre de la secuencia acompañado de la función NextVal, de forma que, siguiendo con el ejemplo, el campo identificador del punto debería llevar el siguiente valor: *incremento_id_punto.NextVal*.

En definitiva, tanto con secuencias como con disparadores, los resultados obtenidos serán los mismos ya que se realizan operaciones similares.

Con el objetivo de ilustrar las inserciones más relevantes del sistema, resulta interesante conocer cómo se insertan los valores espaciales, es decir, los atributos de tipo SDO_GEOMETRY.
En el caso de la inserción de un punto, el INSERT completo se muestra en el Ejemplo 14.

```
INSERT INTO PUNTO VALUES (‘’,
SDO_GEOMETRY(2001,NULL,SDO_POINT_TYPE(posX, posY,
NULL),NULL,NULL), ‘descripcion’);
```

Ejemplo 14. Inserción en la tabla Punto

La tabla Punto tiene como atributos el identificador, la localización geo-espacial del mismo y una descripción. Para almacenar esta información se crea una cláusula INSERT cuyos valores serán los datos a almacenar. En primer lugar, como se avanzó anteriormente, el identificador está en blanco, pues será el disparador correspondiente el que lo rellene.

Para almacenar el punto se utiliza el tipo SDO_GEOMETRY.

- SDO_GTYPE = 2001. Este valor indica que geométricamente se desea almacenar un punto (el 2 indica dos dimensiones y el 1 que es un punto).
- SDO_SRID = NULL. Es el sistema de referencia espacial. No es preciso añadir ningún valor, ya que el valor NULL indica un sistema de referencia cartesiano que es el que se desea utilizar en esta aplicación.
- SDO_POINT = SDO_POINT_TYPE(posX, posY, NULL). Este atributo contendrá las coordenadas del punto (posX y posY), y un NULL en el tercer valor que indica que en este caso no se tendrá coordenada Z.

- SDO_ELEM_INFO y SDO_ORDINATES, los dos últimos valores, están a NULL pues las coordenadas del punto ya han sido dadas y no precisa de más valores. Estos atributos se utilizan a la hora de almacenar polígonos o líneas (como se verá a continuación), pues es en estos donde se guardan las coordenadas que formarán este tipo de elementos e información extra necesaria para que Oracle sepa como unir dichas coordenadas.

Por último, el valor de la descripción debe estar entre comillas simples, pues es de tipo caracter.

Otra inserción ilustrativa es la correspondiente a los enlaces, tal y como se muestra en el Ejemplo 15.

.

```
INSERT INTO ENLACE VALUES (‘’, idNodo1, idNodo2, distancia,
SDO_GEOMETRY(2002,NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,4,2,1,2,1),
SDO_ORDINATE_ARRAY(coord1X, coord1Y, coord2X, coord2Y)));
```

Ejemplo 15. Inserción en la tabla Enlace

Para los enlaces el identificador se incrementa automáticamente gracias al disparador. Los identificadores de los nodos y la distancia, la cual se obtiene gracias a una consulta, se añaden sin necesidad de comillas simples pues están formados por dígitos numéricos. La forma del enlace (línea) es un atributo de tipo SDO_GEOMETRY que tiene las siguientes características:

- SDO_GTYPE = 2002. Este valor indica que geométricamente se desea almacenar una línea (el primer 2 indica dos dimensiones y el segundo 2 indica una o más líneas o segmentos).

- SDO_SRID = NULL. Es el sistema de referencia espacial. No es preciso añadir ningún valor pues, como se indicó anteriormente, se usará el sistema cartesiano.

- SDO_POINT = NULL. No se almacena un punto por lo que el valor de este atributo es NULL.

- SDO_ELEM_INFO = (1,4,2, 1,2,1). Los tres primeros dígitos indican que el elemento es una línea compuesta por dos subelementos. Los siguientes tres dígitos indican que la línea es recta, es decir, no tiene forma curva.

- SDO_ORDINATES = (coord1X, coord1Y, coord2X, coord2Y). Este atributo contiene los valores de las coordenadas que conforman la línea, es decir, valores numéricos. En la aplicación estos valores se pasan por parámetro.

Por último, cabe destacar la definición del SRE, es decir, la inserción en la tabla USER_SDO_GEOM_METADATA de los campos geoespaciales. Para ello se debe especificar la tabla, el atributo geoespacial y las características del mismo. Un ejemplo se presenta en el Ejemplo 16, en la que el atributo espacial indica la localización de los puntos (tabla PUNTO y columna GEO_Localizacion) en la que se supone que el escenario tendría unas dimensionas de 30x30 metros y en la que se tomaría el valor de precisión 0.05, que indica que la distancia más próxima a la que pueden estar dos puntos es 5 centímetros.

```
INSERT INTO user_sdo_geom_metadata
(TABLE_NAME, COLUMN_NAME, DIMINFO,SRID)
VALUES ('PUNTO','GEO_Localizacion',
SDO_DIM_ARRAY( -- 30X30 grid
SDO_DIM_ELEMENT('X', 0, 30, 0.05),
SDO_DIM_ELEMENT('Y', 0, 30, 0.05)),
NULL – SRID
);
```

Ejemplo 16. Inserción en la tabla USER_SDO_GEOM_METADATA

Borrado de registros

Normalmente, el sistema inserta nuevos elementos, pero también puede darse el borrado de algunos de ellos, en concreto de puntos, nodos y enlaces.

Salvo casos excepcionales, que se estudiarán más adelante en el apartado de implementación de la aplicación, los borrados se llevan a cabo mediante el identificador, de forma que, por ejemplo, para borrar un enlace la sentencia se muestra en el Ejemplo 17.

```
DELETE FROM ENLACE WHERE Id_enlace = idEnlace;
```

Ejemplo 17. Borrado de un Enlace

La cláusula DELETE está seguida del nombre de la tabla en la cual se borrará el registro, y éste será el que tenga el idEnlace igual al pasado por parámetro.

Consultas relevantes

Además de la inserción y del borrado de registros, el sistema realizará algunas consultas con el objetivo de obtener datos de interés para llevar a cabo diversas funcionalidades. Las más relevantes están relacionadas con los atributos de tipo SDO_GEOMETRY, y son las que se indican a continuación:

- Obtención de las coordenadas (punto) de un nodo. Para ello se utiliza la SELECT que se muestra en el Ejemplo 18.

```
SELECT n.Localizacion.SDO_POINT.X, n.Localizacion.SDO_POINT.Y FROM
NODO n;
```

Ejemplo 18. Consulta que obtiene las coordenadas de un Nodo

Se utiliza un alias (n) de la tabla Nodo que será sobre la que se realiza la consulta. Para conseguir las coordenadas del punto es preciso acceder al valor de las coordenadas X e Y del atributo Localización.

- Otra consulta interesante, anteriormente mencionada, es la obtención de la distancia. Para ello se utiliza la función SDO_GEOM.SDO_DISTANCE que calcula la distancia entre dos objetos geométricos. A esta función se le deben pasar tres valores: los dos objetos geométricos de los cuales se desea conocer la distancia entre ambos (deben ser de tipo SDO_GEOMETRY) y el valor de la tolerancia (tol). Este número representa la mínima distancia a la que deben estar dos puntos para ser considerados distintos, por ejemplo para dar cabida a errores de redondeo. Se mide en metros, de forma que si se indica un valor 100 de tolerancia significa que en el espacio geométrico utilizado los objetos separados por 100 metros, o menos, se pueden considerar iguales. Por tanto, para este caso, se ha decidido que un valor de tolerancia 0.05 es adecuado,

pues dicho valor de tolerancia indica que a partir de 5 centímetros entre objetos, éstos son considerados diferentes.

La consulta que ilustraría la distancia entre dos puntos se muestra en el Ejemplo 19.

```
SELECT  SDO_GEOM.SDO_DISTANCE (p.Localizacion, p1.Localizacion,
0.05) PUNTO p, PUNTO p1 WHERE (p.Id_punto <> p1.Id_punto)
```

Ejemplo 19. Consulta para obtener la distancia entre dos puntos

Las consultas que se han analizado en este apartado pueden verse modificadas en la implementación en función de los datos que se precisen para la funcionalidad correspondiente. No obstante, existe una gran variedad de métodos de inserción, borrado y obtención de datos que serán explicados en el siguiente apartado.

4.2. Implementación de la aplicación

En el siguiente apartado se explica la implementación de las clases que forman aplicación. En el caso de la interfaz gráfica, se ha utilizado Swing, una biblioteca gráfica de Java. En el anterior capítulo se ha explicado el diagrama de clases de la aplicación, por lo que este apartado analiza y detalla todas las clases y métodos implementados. Previamente, para presentar de forma general esta información, la Figura 33 muestra el diagrama de clases completo.

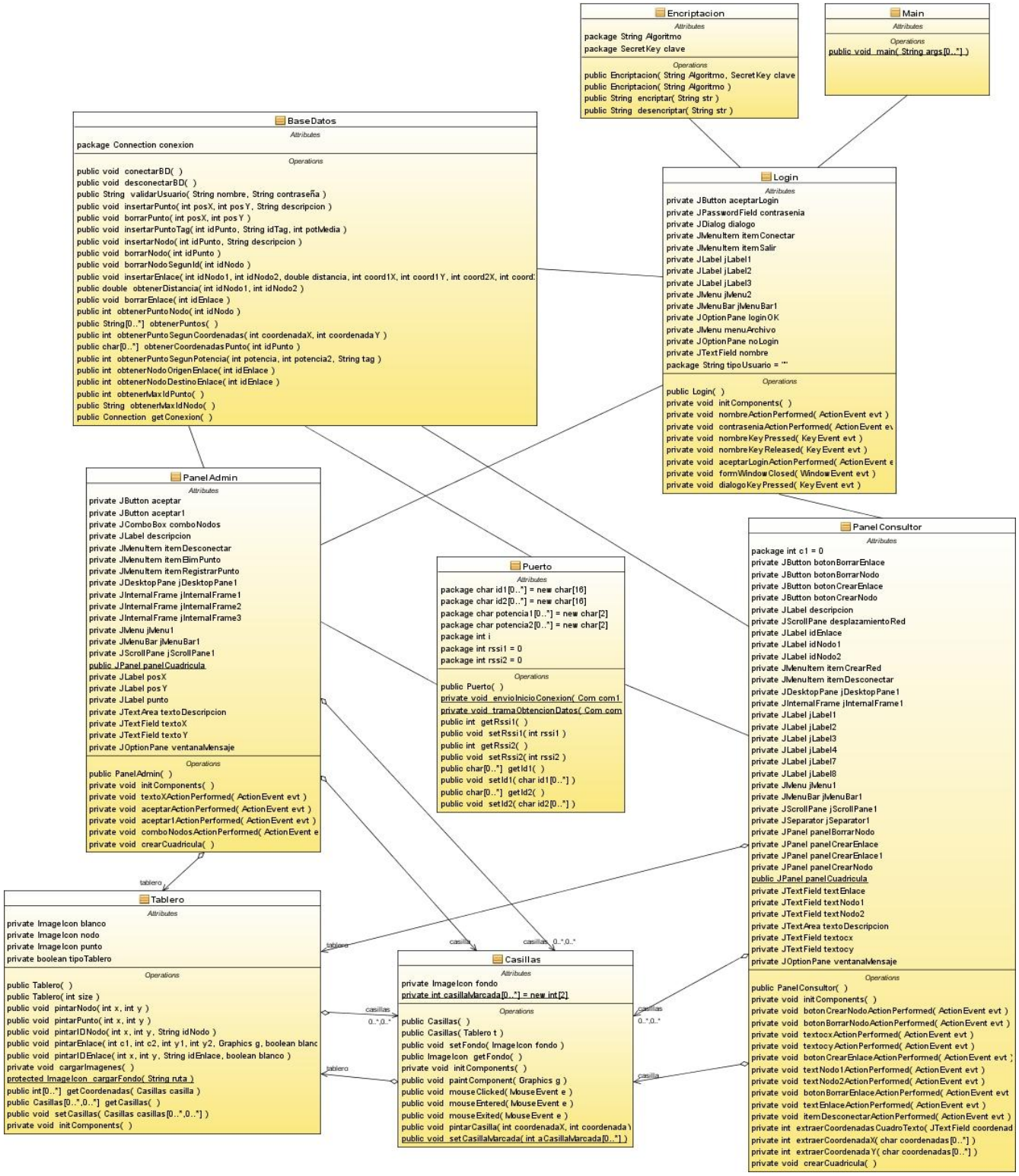


Figura 33. Diagrama de clases completo de la aplicación

4.2.1 Main

Esta clase es el punto de partida de la aplicación, por lo tanto, la primera clase que ejecuta el sistema. Básicamente, se encarga de llamar a la clase Login, que mostrará el primer panel de interfaz que visualizará el usuario.

4.2.2 Puerto

Como ya se ha explicado anteriormente, la experimentación y obtención de datos de la arquitectura RFID que se va a utilizar precisa conectar el lector al puerto serie RS-232 (COM) para que así éste pueda interactuar con los tags disponibles.

Para llevar a cabo este proceso se ha creado una clase en lenguaje de programación JAVA, que por medio de la librería GiovynetSerialPort [46], permite realizar conexiones con los puertos serie.

Esta librería contiene funciones necesarias en el proyecto, ya que las mismas indican los puertos COM disponibles, establece conexiones con los mismos, cierra dichas conexiones, envían y reciben tramas, etc.

Para este caso concreto, una vez establecida la conexión con el puerto COM correspondiente, se especifica la tasa de baudios (número de unidades de señal por segundo, se mide en bits por segundo: bps) a utilizar. Por defecto, el lector trabaja con 9600 bps pero después de las pruebas realizadas, la obtención de datos sólo se realiza correctamente funcionando con 115200 bps.

Llegados a este punto, el lector ya está disponible para "escuchar" a las tags. Por tanto, se debe implementar dicho intercambio de información.

Para conocer con exactitud las tramas que se envían y se reciben, con el objetivo de simularlas en la clase JAVA (implementarlas), se han utilizado dos aplicaciones en paralelo:

- Una de ellas, es el software oficial del lector RFID: Xtive [47]. Esta aplicación muestra por pantalla la obtención de datos que se pretende conseguir con la implementación desarrollada. No obstante, se descarta la utilización de este

software para centralizar la aplicación en el mismo aplicativo y mismo lenguaje de programación.

- A su vez, se ha utilizado también la aplicación Portmon, software que monitorea los puertos serie, de tal forma que muestra por pantalla las tramas que se envían y reciben y por ende, la información que se intercambian el puerto y el lector.

El último proceso mencionado es el que se ha implementado en la clase Java. Básicamente el intercambio de datos se basa en la comunicación entre el puerto serie y el lector RFID, a través del conector USB. El lector recibe las tramas que le envía el puerto serie, obtiene la información de las tags que localiza, y finaliza enviando al puerto las tramas con dicha información. El intercambio de datos de las tramas se realiza caracter por caracter y en hexadecimal, y la información que contiene cada uno se explica a continuación.

En primer lugar se envía la primera trama de conexión entre el puerto serie y el lector RFID. Cabe destacar que el comienzo de trama, cabecera, se expresa con 0x01 y el fin de trama se expresa con el retorno de carro (0x0D). A partir de entonces, y después de la lectura de datos, se envían y reciben las tramas de obtención de datos, puesto que el lector y las tags se comunican constantemente.

Por tanto, a partir de la segunda trama se lleva a cabo el tratamiento de los datos. Los dos primeros caracteres que se reciben se desestiman puesto que no son datos identificativos de la tag. Los 16 siguientes caracteres conforman el ID de la tag, y los 2 siguientes la potencia recibida (RSSI). Se siguen recibiendo caracteres sin información relevante hasta un nuevo retorno de carro.

Este proceso de tratamiento de datos se repite tantas veces como tags haya, por lo que se debe enviar esta segunda trama en función de las tags existentes.

Además, se ha podido observar que, en ocasiones, el lector lee dos veces la misma tag. Este hecho también se ha controlado, por medio de bucles, de forma que cuando se dé el caso vuelva a repetir la lectura de tag, de forma que siempre lea los dos tags que participan en el modelado.

Se finaliza todo el proceso cerrando la conexión con el puerto COM correspondiente. Un ejemplo de este proceso queda representado en la Figura 34.

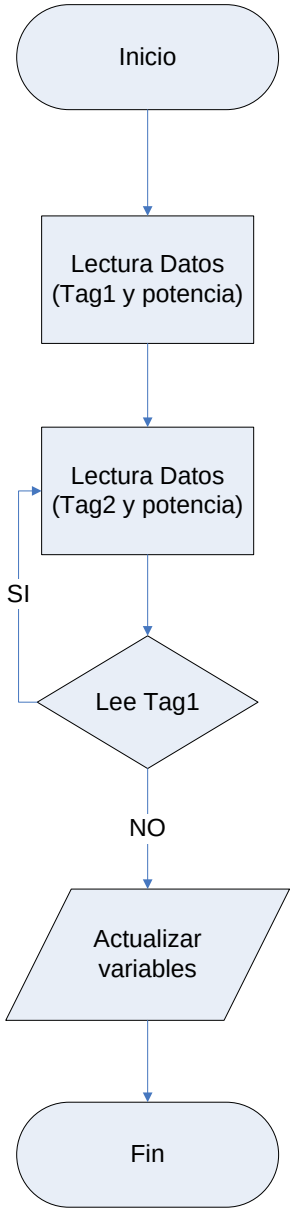


Figura 34. Proceso de lectura de las etiquetas RFID

En dicha Figura 34 se observa el inicio del proceso y el comienzo de lectura de datos de un primer tag. Una vez leídos el identificador y potencia del primer tag (tag 1) se procede a seguir leyendo otra tag, que en principio, debería ser diferencia (tag 2). En cambio, si en lugar de leer un tag diferente, se lee el mismo que en el paso anterior, se debe volver a la lectura de segundo tag pues es evidente que sino el proceso no sería

correcto pues no se habrían leído los dos tags que participan en el sistema. En el caso de que se haya leído un segundo tag diferente, se acutalizan las variables con los datos obtenidos y se procede a finalizar este proceso.

4.2.3 BaseDatos

Esta clase es la encargada de gestionar todas las operaciones relacionadas con la base de datos. Para ello, se necesita añadir el JDBC de Oracle al proyecto, que es la interfaz que propociona Java con el fin de conectarse con Oracle, y ejecutar consultas y órdenes sobre dicho SGBD.

Para poder conectar contra la base de datos se precisan unos parámetros de conexión que hay que configurar. En primer lugar, el driver de conexión, o controlador de la base de datos Oracle, y en segundo lugar la URL de conexión, que debe incluir el nombre y puerto de la base de datos así como el nombre de usuario y contraseña de conexión. De forma general, si fueran cadenas de texto (String), deberían tener el siguiente aspecto:
url = “jdbc:oracle:thin:nombreusuario/contraseña@localhost:1521:XE”.

driver = “oracle.jdbc.driver.OracleDriver”.

No obstante, con el objetivo de hacer más sencilla la configuración y evitar las modificaciones en el propio código de la clase, se ha decidido incluir estos parámetros en un fichero de propiedades llamado *Parametros.properties*. En este fichero se incluyen los valores de las variables driver, URL, user (usuario de la base de datos) y pass (contraseña del usuario de la base de datos).

Estas variables se utilizarán en los métodos de esta clase para establecer conexiones a la base de datos y poder realizar operaciones sobre ella.

Antes de analizar los métodos que alberga esta clase, hay que tener en cuenta, pues son muy importantes y se utilizan constantemente en los métodos implementados, las siguientes interfaces:

- Connection: es el objeto que realizará una conexión física a la base de datos a la que sea preciso conectarse. Esta conexión permite realizar operaciones sobre ella.

- Statement: permite crear una sentencia SQL utilizando el objeto Connection.
- ResultSet: permite guardar el resultado de una consulta en función del objeto Statement usado.

Vistos estos interfaces, a continuación se analizan cada uno de los métodos que esta clase:

- conectarBD(): este método es el encargado de realizar la conexión a la base de datos. Hay que tener en cuenta que, como se indicaba anteriormente, los parámetros de conexión se han incluido en un fichero de propiedades, por lo que se ha utilizado la clase ResourceBundle para recoger dichos valores del fichero. Para ello, a una variable string se le asigna el nombre de fichero de propiedades (nombre del paquete más nombre del fichero). Además, se instancia la clase con la información extraída en la variable anterior. El Ejemplo 20 muestra el código utilizado.

```
String properties = "BaseDatos.Parametros";
ResourceBundle rb = ResourceBundle.getBundle (properties);
```

Ejemplo 20. Acceso a fichero de propiedades con ResourceBundle

Una vez realizado este paso, ya se está en disposición de establecer la conexión. En primer lugar, se carga el controlador de la base de datos, es decir, se carga el driver. En el Ejemplo 21 se presenta un ejemplo del código.

```
try {
    //Cargar clase de controlador de base de datos
    Class.forName(rb.getString("driver"));
} catch (ClassNotFoundException exc) {
    System.out.println("Error al cargar el driver de la BD" + " : " +
exc);
    exc.printStackTrace();
}
```

Ejemplo 21. Carga del driver de la Base de Datos

Si la carga se ha realizado correctamente, crea el objeto de conexión con la base de datos en función de los parámetros configurados en el fichero de propiedades citado anteriormente.. Un ejemplo de la línea de código se presenta en el

Ejemplo 22. Aunque en dicho ejemplo no se ha incluido, este código también contiene el control de excepciones *try-catch*.

```
conexion = DriverManager.getConnection(rb.getString("url"), rb.getString("user"),
rb.getString("pass"));
```

Ejemplo 22. Objeto de conexión a la Base de Datos

- desconectarBD(): cierra la conexión con la base de datos.
- validarUsuario (String nombre, String contraseña): método encargado de validar los datos del usuario. Para ello realiza una consulta a la tabla USUARIO en función de los valores pasados por parámetro. Este método devuelve el tipo de usuario que accede a la aplicación, si lo hubiere.
- insertarPunto(int posX, int posY, String descripcion): método encargado de insertar un punto en la base de datos, en la tabla PUNTO. Las coordenadas del mismo se pasan por parámetro. Tal y como se indicó en el apartado de implementación de Base de Datos se debe realizar un INSERT teniendo en cuenta que la posición (localización) del punto es un atributo geo-espacial. Para realizar la inserción correspondiente se utiliza el método de base de datos *executeUpdate()* en lugar del método *executeQuery(),* que es el que se usa para las consultas.
- borrarPunto (int posX, int posY): método encargado de borrar un punto de la base de datos en función de las coordenadas pasadas por parámetro. Como el borrado se realiza a partir de estas coordenadas, se accede a las coordenadas X e Y del atributo Localización del punto y si coinciden se realiza el borrado de esa tupla. En el caso de que existan varios puntos con esas coordenadas, se eliminará todos ellos.
- insertarPuntoTag (int idPunto, String idTag, int potMedia): inserta en la tabla intermedia PUNTO_TAG un registro en función de los datos pasados por parámetro. Cabe recordar que el valor de la potencia media viene dado por el cálculo del valor medio de las potencias obtenidas en tres ocasiones. Este proceso se realiza antes de la llamada a este método, y el valor obtenido se le pasa a éste por parámetro.

- insertarNodo (int idPunto, String descripcion): inserta un nodo en la base de datos. Éste estará situado sobre un punto, de ahí la necesidad de indicar el valor del identificador del punto por parámetro.
- borrarNodo (int idPunto): borra un nodo de la base de datos. Al igual que en el método anterior, el borrado viene dado por el valor del identificador del punto pasado por parámetro. Si coincide este valor con el del atributo Id_punto se borra el registro correspondiente.
- borrarNodoSegunId (int idNodo): este método borra el nodo de la forma más habitual, en función del propio identificador del nodo.
- insertarEnlace (int idNodo1, int idNodo2, double distancia, int coord1X, int coord1Y, int coord2X, int coord2Y): inserta un enlace en la base de datos. Para ello precisa el valor de los identificadores de los nodos que forman el enlace, la distancia existente entre ambos (es un valor de un atributo geo-espacial) y se precisa también el valor de las coordenadas de cada nodo, pues dichos valores se incluyen en el atributo SDO_ORDINATE (Sdo_Ordinate_Array) del atributo Forma (SDO_GEOMETRY).
- borrarEnlace (int idEnlace): borra el enlace en función del identificador pasado por parámetro.
- obtenerDistancia(int idNodo1, int idNodo2): obtiene la distancia entre los dos nodos pasados por parámetro. La forma de obtener la distancia se explicó, desde el punto de vista de la base de datos, en el apartado anterior.
- obtenerPuntoNodo (int idNodo): método que obtiene el identificador del punto asociado al nodo pasado por parámetro.
- obtenerPuntos(): devuelve en un array (ArrayList) las coordenadas de todos los puntos almacenados en la base de datos.
- obtenerPuntoSegunCoordenadas(int coordenadaX, int coordenadaY): devuelve el identificador del punto cuyas coordenadas tengan los valores pasados por parámetro. Para realizar la comparación entre estos valores y los valores almacenados en la base de datos se accede a las coordenadas X e Y del atributo *Localizacion* del punto.
- obtenerCoordenadasPunto(int idPunto): devuelve una variable de tipo char[] que contiene las coordenadas X e Y del punto pasado por parámetro. Se ha optado por el tipo char con el fin de hacer más sencillo el algoritmo, sobre todo a la hora

de parsear los datos a otro tipo de datos. Una posible alternativa sería usar el tipo de dato Double.

- obtenerPuntoSegunPotencia(int potencia1, int potencia2, String tag): método que devuelve el identificador del punto según las potencias pasadas por parámetros. En el capítulo de Pruebas se detalla en profundidad la forma en la que funciona este método.
- ObtenerNodoOrigenEnlace(int idEnlace): método que obtiene el valor del identificador del nodo de origen del enlace.
- ObtenerNodoDestinoEnlace(int idEnlace): método que obtiene el valor del identificador del nodo de destino del enlace.
- obtenerMaxIdPunto(): obtiene el máximo valor del identificador de los puntos almacenados en la base de datos.
- obtenerMaxIdNodo(): obtiene el máximo valor del identificador de los nodos almacenados en la base de datos.
- getConexion(): método que permite retornar la conexión haciendo un get de la variable de conexión.

4.2.4 Tablero y Casillas

Las clases Tablero y Casillas se incluyen dentro del paquete Utilidades y son las encargadas de pintar la cuadrícula para crear la red. Sobre ella se pintarán los nodos y los enlaces que el usuario crea preciso.

Tablero es una clase que extiende del panel Consultor (JPanel). El tablero creado será cuadrado, es decir, el eje X y el eje Y tendrán el mismo número de casillas. Para ello, a la hora de crear la instancia de tablero, y gracias al método público Tablero(), se le pasa por parámetro el tamaño del mismo. Una vez cargados los componentes por el método *initComponents()*, se dispone el tablero de forma cuadrada y se llama al método *cargarImagenes()* que se encarga de cargar las imágenes para pintar un nodo (nodo.gif) y para pintar un hueco en blanco en cada casilla (blanco.gif), es decir, de cargar estas imágenes en dos variables de tipo *ImageIcon*. Para realizar este paso se llama al método *cargarFondo (String ruta)* al que se le pasa por parámetro la ruta donde se encuentran

dichas imágenes. De esta forma ya se tienen disponibles las imágenes que se van a utilizar para pintar en el tablero.

Una vez cargadas las imágenes, se crea un array de Casillas de dos dimensiones, en el que se crearán las casillas que conforman el tablero y además, en cada una de ellas, se cargarán la imagen blanco.gif. Estas casillas tienen un tamaño específico de 20 píxeles dado por el método setBounds().

Los demás métodos de la clase Tablero son los siguientes:

- pintarNodo (int x, int y): se encarga de pintar la imagen nodo.gif (utilizando la clase graphics) en la casilla correspondientes (las coordenadas de la misma se pasan por parámetro).
- pintarIDNodo (int x, int y, String idNodo): similar al método anterior salvo que en este caso se pinta en la casilla correspondiente el valor del identificador del nodo de forma que se visualice en la casilla (en el centro) la imagen del nodo y su id.
- pintarEnlace (int c1, int c2, int y1, int y2, Graphics g): método que se encarga de pintar una línea (enlace) entre las dos casillas cuyas coordenadas se pasan por parámetro.
- getCoordenadas (Casillas casilla): método que obtiene las coordenadas de la casilla pasada por parámetro.
- getCasillas(): obtiene el array de dos dimensiones de casillas.
- setCasillas(Casillas[][] casillas): establece el valor del array de dos dimensiones de las casillas en función de la variable pasada por parámetro.

La clase Casillas participa en la clase anterior (y extiende de un elemento JPanel), pues el tablero estará compuesto de sus casillas correspondientes.

Cabe destacar que en un principio se analizó la posibilidad de permitir al usuario añadir nodos gracias a las pulsaciones (clicks) de ratón. Esta funcionalidad resultaba técnicamente posible gracias a la implementación de métodos y eventos de ratón, pero

recoger los datos, tratarlos y almacenarlos en la base de datos se antojaba muy complejo, por lo que finalmente se decidió utilizar campos de texto rellenables por el usuario para conseguir la información susceptible de almacenar y pintar. No obstante, esta funcionalidad queda recogida en forma de comentarios en el código por lo que puede resultar interesante para posibles líneas futuras.

El método public Casillas (Tablero t) inicializa los componentes gracias al método initComponents() y asigna el tablero a una variable del tipo de dicha clase.

Los demás métodos de la clase Casillas son los siguientes:

- getFondo(): obtiene la imagen de fondo.
- setFondo (ImageIcon fondo): se encarga de asignar a una variable la imagen de fondo pasada por parámetro.

4.2.5 Encriptación

La clase Encriptacion se incluye también dentro del paquete de Utilidades y se utiliza para encriptar y desencriptar las contraseñas de los usuarios que acceden a la aplicación. Para realizar este proceso se han utilizado las clases KeyGenerator [48] y Cipher [49] que forman parte, entre otras clases, del framework de criptografía JCE (Java Cryptography Extension) [50].

La clase KeyGenerator es usada para generar claves secretas que se utilizan en algoritmos simétricos. Se instancia a través del método *getInstance()* indicando el algoritmo que se desea usar para generar las claves. Para este proyecto se ha utilizado el algoritmo *DES*. A su vez, para crear una clave se debe crear un objeto de la clase SecretKey (para almacenar la clave) y usar el método *GenerateKey()*.
En el Ejemplo 23 se refleja la línea de código utilizada para este proceso:

```
SecretKey key = KeyGenerator.getInstance("DES").generateKey();
```

Ejemplo 23. Creación de un objeto de la clase SecretKey con keyGenerator

La clase Cipher se utiliza para cifrar con algoritmos de clave simétrica. También se utiliza el método *getInstance()*. A continuación se detallan los métodos de encriptación y desencriptación, con los que se entrará algo más en detalle con esta clase Cipher:

- encriptar (String str): este método encripta la cadena de texto pasada por parámetro, en este caso la contraseña. Para ello, en primer lugar se pasa a bytes la cadena de texto. A continuación se crea un objeto de la clase Cipher con el algoritmo de cifrado elegido y se configura con los parámetros deseados. Por último se realiza el cifrado y se pasa a Base64 para obtener el String resultante. En el Ejemplo 24 se muestra código de ejemplo de este proceso.

```
byte[] cadenaByte = str.getBytes("UTF8");
Cipher cifrar = Cipher.getInstance(Algoritmo);
cifrar.init(Cipher.ENCRYPT_MODE, clave);
byte[] enc = cifrar.doFinal(cadenaByte); // ciframos
// ---------- Pasamos aBase64 para obtener un String
        s = new sun.misc.BASE64Encoder().encode(enc);
```

Ejemplo 24. Funcionalidad de encriptación

- desencriptar (String str): este método desencripta la cadena de texto pasada por parámetro. El proceso es el contrario al de encriptación en el sentido de que también hay que crear el objeto de la clase de datos Cipher, aunque en este caso se debe configurar para realizar la operación de desencriptación. En la figura X se tiene un ejemplo.

```
// ------------ Obtenemos los Bytes del String encriptado
        byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);
Cipher descifrar = Cipher.getInstance(Algoritmo);
descifrar.init(Cipher.DECRYPT_MODE, clave);          byte[]
cadenaByte = descifrar.doFinal(dec);
// ------------- Volvemos a pasarlo a forma de cadena.
        s = new String(cadenaByte, "UTF8");
```

Ejemplo 25. Funcionalidad de desencriptación

Cabe destacar que para instanciar a esta clase Encriptacion es necesario crear el objeto indicando el algoritmo que se pretende utilizar. En el Ejemplo 26 se muestra este caso, utilizando el algoritmo DES.

```
Encriptacion encrip = new Encriptacion ("DES");
```

Ejemplo 26. Creación de un objeto de la clase Encriptacion con DES

4.2.6 Login

La clase Login es llamada por la clase Main y se encarga de presentar al usuario el primer panel de interacción con el sistema. Este panel, gracias a la utilización de la paleta de objetos Swing, permite diseñar la interfaz gráfica al gusto del programador de forma totalmente gráfica. Al igual que en el resto de clases, se han estudiado muchas de las propiedades que contienen los elementos gráficos.

El método principal de la clase, es decir, el método *Login()*, que es de carácter público, se encarga de inicializar todos los componentes y elementos gráficos que forman parte de esta clase, gracias a la llamada al método *initComponents()*. Este método, internamente, contiene los valores y el diseño de todos los elementos gráficos añadidos en la clase, de forma que los menús, labels (etiquetas), campos de texto, etc., así como sus propiedades, se inicializarán en él.

Una vez inicializados los componentes se hace visible el panel a pantalla completa con un menú que tiene la opción Archivo. Este punto de menú tiene dos ítems: Conectar y Salir.

El ítem Salir cierra la aplicación y la instancia de la Base de Datos.

El ítem Conectar presenta una ventana de diálogo posicionada en el centro de la pantalla que permite validar los datos (usuario y contraseña) del usuario. Una vez pulsado el botón aceptar desaparece la ventana de diálogo e internamente se llama al método (evento de botón) *aceptarLoginActionPerformed()* que será el encargado de validar estos datos. Para ello, realiza una conexión a la base de datos llamando al método validarUsuario() de la clase BaseDatos .

Una vez validados estos datos se presentan tres opciones:

- Si los datos son erróneos se muestra una ventana con un mensaje de error indicando que los datos introducidos no son correctos por lo que no se valida el usuario. Una vez cerrada esta ventana de confirmación, el usuario se encontraría

ante la situación inicial, teniendo opción de volver a conectarse usando la opción Conectar del menú.

- Si los datos introducidos corresponden a un usuario de perfil Consultor se cerraría el panel y se crearía una instancia de la clase PanelConsultor() que presenta el panel y el menú con las opciones y herramientas disponibles para los usuarios de este perfil. Una ventana de confirmación, con su mensaje correspondiente, informa de este hecho. Cabe destacar la navegabilidad entre clases y paneles gracias a la creación (new()) de la nueva instancia y ocultación (dispose()) de la anterior.

- Si los datos introducidos corresponden a un usuario de perfil Administrador se cierra el panel y se crea una instancia de la clase PanelAdmin() que presenta el panel y el menú con las opciones y herramientas disponibles para los usuarios de este perfil. Una ventana de confirmación, con su mensaje correspondiente, informa de este hecho. Cabe destacar en este caso la navegabilidad entre clases y paneles gracias a la creación (new()) de la nueva instancia y ocultación (dispose()) de la anterior.

4.2.7 PanelAdmin

La clase PanelAdmin se encarga de mostrar en un panel las funcionalidades correspondientes del usuario de perfil Administrador una vez éste se ha validado. Cabe recordar que este usuario se encargará, gracias a funcionalidades disponibles, de modelar el escenario en el que posteriormente se podrá crear la red.

El método principal de la clase es el método *PanelAdmin()*, que es de carácter público y que se encarga de inicializar todos los componentes y elementos gráficos que forman parte de esta clase gracias a la llamada al método *initComponents()*. Este método, internamente, contiene los valores y el diseño de todos los elementos gráficos añadidos en la clase, de forma que los menús, labels (etiquetas), campos de texto, etc., así como sus propiedades, se inicializarán en él.

Una vez inicializados los componentes se hace visible el panel a pantalla completa con un menú que tiene la opción Archivo. Este punto de menú tiene tres ítems: Registrar Punto, Eliminar Punto y Desconectar. A su vez, en el propio panel, se dibuja una

cuadrícula con casillas en blanco que servirá para representar los puntos que se vayan creando.

El ítem Desconectar cierra la ventana actual y vuelve al panel principal de Login, de forma que cierra la sesión con el usuario actual.

El ítem Registrar Punto presenta una ventana interna que permite introducir los datos necesarios para registrar un punto: las coordenadas X e Y, y una descripción. Cabe recordar que el lector RFID debe estar conectado al PC para poder leer las etiquetas RFID existentes en la sala (escenario). De esta forma, el punto es modelado por el usuario gracias a las coordenadas introducidas y las potencias obtenidas por dichas etiquetas.

Para validar los datos introducidos se debe pulsar el botón aceptar, que a su vez llama al método *aceptarLoginActionPerformed()*. Éste crea una nueva variable (constructor) de la clase Puerto, que internamente obtiene la potencia media correspondiente para el punto que se desea registrar. Se ha decidido que la potencia media se obtenga en base a tres mediciones, es decir, el sistema, dentro de esta clase PanelAdmin, repite en tres ocasiones la lectura de los tags y de sus respectivas potencias (llamada a la clase Puerto), y finalmente calcula la media de las medidas obtenidas. Naturalmente, esta validación de datos debe almacenarse en base de datos por lo que se crea una instancia de la misma llamando a los métodos *insertarPunto()* e *insertarPuntoTag()*

Finalmente, si el proceso ha sido correcto, se muestra una ventana de confirmación en la que se indica que los datos se han guardado correctamente y el punto se representará en la cuadrícula.

El ítem Borrar Punto presenta una lista desplegable en la que se recogen todos los puntos existentes mostrados por coordenadas (x, y). Estos puntos se muestran gracias a una lista, recorrida por un iterador (iterator). Cabe destacar que esta lista desplegable se recarga (se eliminan todos los datos y se vuelven a recuperar) cada vez que se inicia el panel Administrador con el objetivo de evitar información duplicada. Esta funcionalidad consulta la base de datos para obtener los puntos almacenados en ella, y a continuación los presenta en el item desplegable. Además, cada vez que se borra un punto en esta pantalla el item se recarga eliminando el punto correspondiente ya que éste no se encontrará en la base de datos.

Una vez seleccionado el punto a borrar en dicha lista desplegable se debe pulsar el botón aceptar para que se realice el borrado, llamando entonces al método *aceptar1LoginActionPerformed()*. Éste obtiene las coordenadas del punto a borrar parseando las mismas para convertirlas a tipo int. Gracias a las coordenadas y a la conexión con la base de datos se está en disposición de borrar el punto, gracias al método borrarPunto() de la clase BaseDatos.

Por último, si el proceso ha sido correcto, se muestra una ventana de confirmación en la que se indica que los datos se han borrado correctamente y el punto quedará borrado de la cuadrícula.

4.2.8 PanelConsultor

La clase PanelConsultor es visualmente similar a la anterior. Se encarga de mostrar en un panel las funcionalidades correspondientes del usuario de perfil Consultor una vez este se ha validado. Cabe recordar que este usuario se encargará, gracias a las funcionalidades disponibles, de crear una red compuesta por nodos y enlaces. Para ello, se creará un tablero cuadrado, de 30 posiciones, por lo que se precisa de variables de las clases Tablero y Casillas. Se ha optado por un tablero de 30 posiciones porque visualmente ofrece una cantidad de casillas interesante para modelar un escenario. En cuanto al uso de un tablero con casillas, se debe a que el mismo dota al sistema de un mayor impacto visual y de una funcionalidad más refinada para el modelado de escenarios por lo que se estima que debe ser un aspecto interesante a estudiar en líneas futuras.

El método principal de la clase, es decir, el método *PanelConsultor()*, que es de carácter público, se encarga de inicializar todos los componentes y elementos gráficos que forman parte de esta clase gracias a la llamada al método *initComponents()*. Cabe destacar que es en este punto donde se crea un panel que albergará el tablero de 30 posiciones.

Una vez inicializados los componentes se hará visible el panel a pantalla completa con un menú que tiene la opción Archivo. Este punto de menú tiene dos ítems: Crear Red y Desconectar.

El ítem Desconectar, al igual que en el panel de perfil Administrador, cierra la ventana actual y vuelve al panel principal de Login, de forma que cierra la sesión del usuario actual.

El ítem Crear Red accede a la principal funcionalidad de este usuario. Se presenta una ventana interna dividida en dos bloques. El bloque izquierdo consta del tablero de 30 posiciones mencionado anteriormente en el que se pintarán los elementos que el usuario indique usando el bloque derecho, en el que el usuario, gracias a los paneles e items facilitados, puede crear y borrar nodos y enlaces indicando, en caso necesario, los datos que desee, como por ejemplo, los ids de los nodos que quiere unir para formar un enlace.
En función de la opción elegida por el usuario, y gracias a los métodos de conexión con la base de datos, el sistema muestra gráficamente en el tablero los elementos correspondientes. Para esta funcionalidad cabe destacar también el uso del hardware pertinente (lector y tarjetas RFID), así como los métodos software que lo sustentan, pues para crear un nodo, tal y como se verá a continuación, el sistema consulta las potencias de las etiquetas RFID y compara dicha información con las potencias de un punto que ya esté creado, para posteriormente situar el Nodo encima de ese Punto. De esta forma, en una única ventana, de manera intuitiva y sencilla, se podrá visualizar en qué posición del tablero se encuentra el usuario, es decir, en qué posición del tablero se ha situado el Nodo.

A continuación, para cada operación posible, se detalla la implementación de los métodos de las mismas:

- Crear Nodo: para crear el Nodo, el usuario no debe introducir ningún valor, en todo caso una descripción si lo desea. Cabe recordar que será el sistema el que, gracias al intercambio de información entre el lector RFID y las etiquetas, obtenga la posición correspondiente según la potencia obtenida en ese momento. Para ello, una vez pulsado el botón Aceptar de este panel, se llama al método botonCrearNodoActionPerformed() que es un evento del componente botón. En primer lugar, se obtiene la potencia, tal y como se ha descrito en la clase Puerto. Una vez obtenida la potencia, se realiza una conexión con la base de datos para conocer qué punto (identificador) del escenario tiene asignado la potencia más

cercana a la que se ha obtenido, ya que generalmente, será complicado obtener exactamente la misma potencia que esté almacenada en la base de datos. Como se verá en el capítulo de Pruebas, se ha utilizado el valor mínimo (min) del valor absoluto (abs) de la diferencia entre el valor de la potencia del experimento y el valor de la potencia media de cada fila de la base de datos: min (ABS (potenciaExp – PotenciaMedia)). De esta forma se consigue que la potencia extraída, ya sea superior o inferior a la potencia media (gracias a la claúsula ABS), sea la más cercana (cláusula MIN).

Una vez se tiene el punto, se recupera la descripción introducida por el usuario y se inserta el nodo en la base de datos. En sentido figurado, se podría decir que el nodo estaría situado encima de dicho punto.

A continuación, este nuevo nodo debe pintarse en el tablero. Para ello, mediante una consulta a la base de datos, se extraen las coordenadas del punto obtenido anteriormente con el objetivo de conocer dónde se ha de pintar dicho nodo en el tablero.
Después de esto, se vuelve a consultar en la base de datos el identificador de este nodo almacenado (que será el último), pues este identificador deberá ser pintado encima del nodo en el tablero.

Por último, con todos los datos necesarios en las variables correspondientes se procede a pintar el nodo y el identificador del mismo con los métodos oportunos de la clase Tablero. Cabe destacar, al igual que a la hora de pintar un enlace, el estudio realizado de la clase Graphics y de sus propiedades, consiguiendo el objetivo de poder pintar elementos exactamente en las posiciones deseadas.

- Borrar Nodo: en este panel el usuario puede borrar el Nodo en función de las coordenadas introducidas por pantalla, es decir, borra el nodo ubicado en unas coordenadas dadas. Cuando el usuario pulsa el botón Aceptar se llama al método, o evento, *botonBorrarNodoActionPerformed()*. Este método recupera el valor introducido en las casillas de texto de las coordenadas para, posteriormente, conectarse a la base de datos y obtener el punto en función de las mismas. Una vez conocido el punto sobre el que está situado el nodo se procede a borrarlo accediendo de nuevo a la base de datos.

Por último, si el proceso ha tenido éxito, se muestra por pantalla un mensaje de confirmación de la operación.

Cabe destacar que también se tiene implementado, para un posible uso futuro, el borrado de nodos facilitando el identificador de los mismos.

- Crear Enlace: este panel se encarga de crear un enlace en función de los dos nodos que se deseen unir. Para ello se debe introducir el identificador de cada uno. Esta operación se lleva a cabo al pulsar el botón Aceptar que llama al método *botonCrearEnlaceActionPerformed()*. Este método, en primer lugar, obtiene los identificadores de los nodos a unir. A continuación, a través de una consulta a la base de datos, calcula la distancia entre ambos nodos, y finalmente se inserta el enlace en la base de datos, gracias a las operaciones de la clase BaseDatos.
  En este momento el enlace está almacenado en la base de datos pero falta pintarlo en el tablero.

  Por cada identificador, mediante accesos a la base de datos, se recuperan las coordenadas de los puntos sobre los que están situados los nodos, con el objetivo de saber qué casillas hay que unir pintando una línea entre ellas. Una vez conocidas las casillas se llama a los métodos correspondientes que, además de pintar el enlace, también pintan el identificador del mismo. Cabe destacar que en este caso se ha utilizado la clase Graphics2D para dibujar la línea, ya que esta clase permitía aumentar el grosor de la misma.

- Borrar Enlace: en este panel el usuario puede borrar el enlace introduciendo el identificador del mismo. El botón aceptar valida la operación, y para ello llama al evento *botonBorrarEnlaceActionPerformed()*. Este método recupera el identificador introducido por el usuario y accede a la base de datos para borrar la tupla del enlace almacenado con dicho id. Posteriormente, para finalizar la operación, el enlace es borrado del tablero.

Por último, internamente, se han implementado métodos privados con el objetivo de encapsular funcionalidades y evitar repetición de líneas de código que se utilizan en varias ocasiones en la clase. Estos métodos son:

- extraerCoordenadasCuadroTexto (JTextField coordenada): obtiene el valor entero del número introducido por el usuario en el campo de texto pasado por parámetro. Para ello se parsea de string a int.

- extraerCoordenadaX(char[] coordenadas): devuelve el valor de la coordenada X de la variable *coordenadas* pasada por parámetro. Para ello, se recupera el valor de la posición 0 de dicha variable (de char a string) y se parsea de string a int.

- extraerCoordenadaY(char[] coordenadas): igual que el método anterior salvo que este devuelve el valor de la coordenada Y, pues recupera el valor de la posición 1 de la variable *coordenadas*.

Independiente de los anteriores, se ha implementado un método que genera el tablero (cuadrícula) cuyo nombre es *crearCuadricula()*. Este método es llamado desde el *ActionPerformed()* de la opción Crear Red, es decir, cuando se genera el panel se llama a este método que crea el tablero según las propiedades implementadas en el método. Éste resulta de vital importancia pues contiene el formato del tablero.

5. Pruebas

A lo largo de estos años de carrera académica, en cada una de las prácticas que precisaran realizar una aplicación, ha sido de vital importancia la elaboración de una batería de pruebas completa y exhaustiva durante y después del proceso de implementación.

Gracias a este proceso de pruebas se puede verificar la implementación y analizar los fallos encontrados así como los métodos y funcionalidades susceptibles de ser mejorados, como por ejemplo la eficiencia, la rapidez o la utilización de recursos por parte de la aplicación.

Aunque durante la fase de implementación se han llevado a cabo pruebas independientes y particulares que verificaban el funcionamiento de cada pequeña funcionalidad de la aplicación, ha sido al final de esta fase de implementación cuando se ha procedido a probar el sistema al completo. No obstante, para este proyecto, adquieren mucha importancia las pruebas del hardware utilizado, es decir, el uso de las etiquetas y el lector RFID y, por tanto, serán estas pruebas las que tengan el mayor peso de la batería de pruebas.

Para probar el hardware, se dispone un escenario, una habitación, en la que en una pared se ubican las dos etiquetas RFID, a una distancia de unos tres metros entre ellas. A una distancia cercana, de unos dos metros, se señalan 9 puntos diferentes que son equidistantes, de forma similar a la Figura 35.

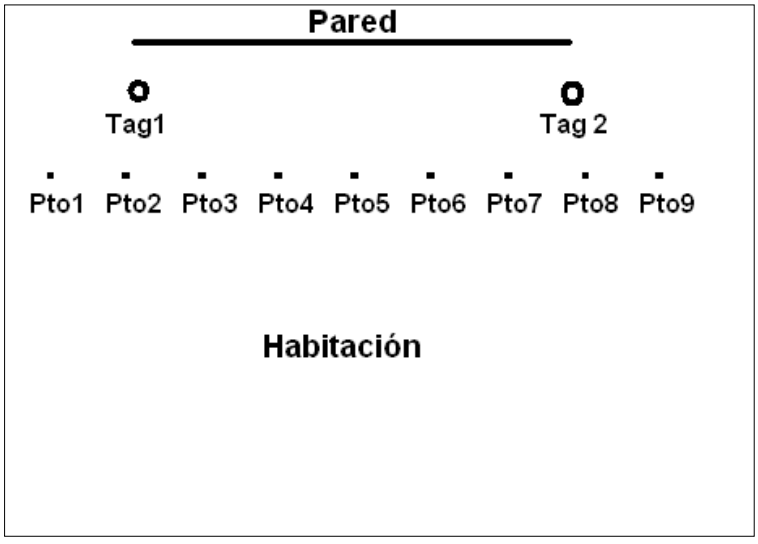


Figura 35. Escenario de pruebas

En primer lugar, en cada punto se sitúa el lector RFID y se mide varias veces, en concreto tres, la potencia recibida por cada tag, para hacer una media. La potencia recogida en cada punto se almacena en la base de datos para, posteriormente, en función de esta información, conocer en qué punto se encuentra el usuario.

Para esta funcionalidad se utiliza una aplicación que contiene una interfaz en la que se puede guardar cada punto (internamente calcula potencia) y también obtener el punto en el que se encuentra el usuario en función de la potencia.

La Figura 36 muestra la interfaz de esta aplicación de pruebas.

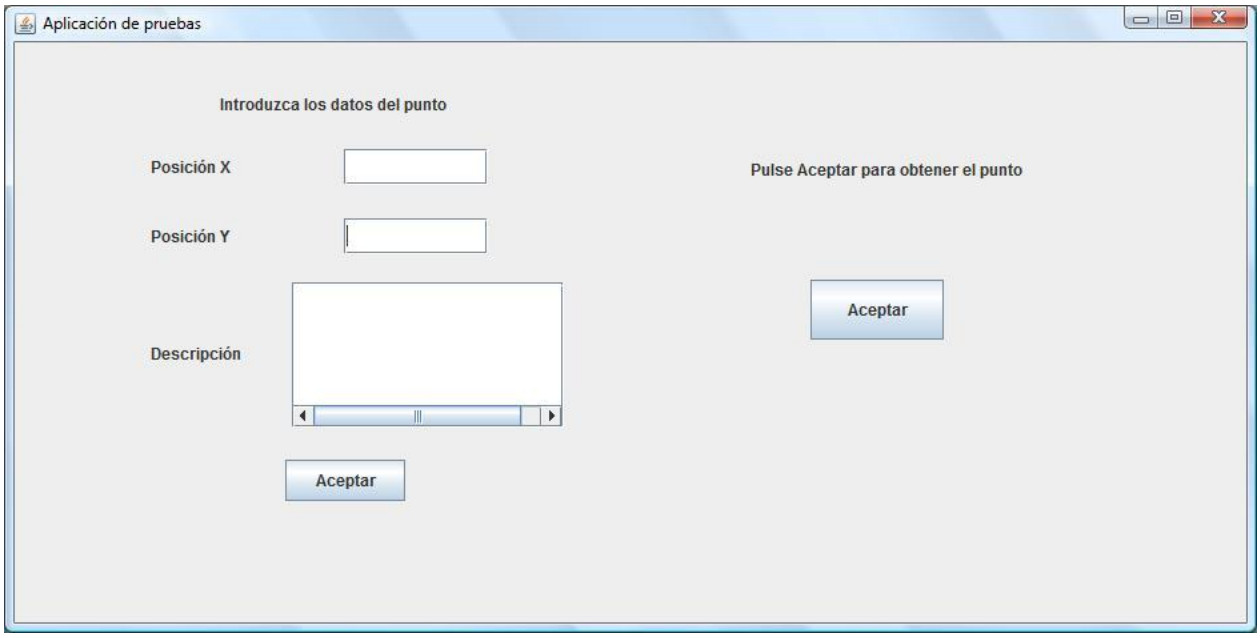


Figura 36. Interfaz de la aplicación de pruebas

Una vez asignadas las potencias en cada punto hay que analizar la probabilidad de acierto que permite saber cuánto de juntos pueden estar los puntos que participan en el escenario, es decir, la granularidad con la que el sistema es capaz de localizar al usuario.

Para ello, se realizan diferentes modelados:

- **Primera prueba**: se modela el escenario únicamente con tres puntos: P1, P5, P9 (precisión 180 cm). Se coloca el lector RFID en cada punto y se apunta si el sistema localiza correctamente el punto. Se repite la prueba 10 veces en cada punto. A su vez, para cada punto, se mide el promedio de acierto, es decir, se suman todas las medidas del punto y se divide entre el número de veces medidas (10). Por último, se calcula la probabilidad de acierto global, es decir, el promedio de la probabilidad de acierto de todos sus puntos.

- **Segunda prueba**: se incluyen los puntos P3 y P7 respecto a la prueba anterior (precisión 90 cm), repitiendo el proceso anterior para los 5 puntos.

- **Tercera prueba**: se incluyen los puntos restantes, es decir, P2, P4, P6 y P8 (precisión 45 cm). Se repite el proceso de las anteriores pruebas. En este caso será una prueba completa pues ya están incluidos todos los puntos en el modelado.

Obtención del punto en función de la potencia

Como se indicaba anteriormente, en primer lugar, por cada punto, se mide la potencia que se recibe y se almacena en la base de datos. Cabe recordar que, al utilizar dos tags, en la base de datos se almacena la potencia recibida asignada al tag correspondiente en dicho punto.

Posteriormente, para conocer el punto en el que está situado el usuario en función de la potencia, el sistema internamente lleva a cabo el siguiente proceso:

- Se obtiene la potencia de las dos tags.
- Se realizan consultas contra la base de datos para obtener el punto. Para estas consultas se utilizan el número de identificación y la potencia de los dos tags. Las consultas se basan en el siguiente funcionamiento: los valores (id del tag y potencia) obtenidos de cada tag leído se comparan con los valores de las filas de la tabla correspondiente, de forma que el punto a obtener será el que tenga mismos identificadores de tag y las potencias sean las más cercanas posibles a las obtenidas previamente.

  Para conseguirlo, se ha utilizado el valor mínimo (min) del valor absoluto (abs) de la diferencia entre el valor de la potencia del experimento y el valor de la potencia media de cada fila de la base de datos: ABS (potenciaExp – PotenciaMedia). Un ejemplo de consulta sería la representada en el Ejemplo 27.

```
SELECT pt.Id_punto FROM Punto_Tag  pt WHERE (pt.Id_tag =
0001000108121106) AND (ABS(130 – pt.Potencia_media)) = (SELECT min
(ABS(130 – pt2.Potencia_media)) FROM Punto_Tag  pt2 WHERE pt.Id_Tag
= pt2.Id_Tag);
```

Donde 130 es la potencia obtenida del tag leído con número de identificador 0001000108121106 . De forma análoga sería para el otro tag.

Ejemplo 27. Consulta para obtener un punto comparando potencias

Por cada tag, se obtendría el punto en función de la potencia, pero en algunos casos el resultado no sería correcto, es decir, se podrían obtener puntos diferentes en función de la potencia de cada tag pues se ha de tener en cuenta

que un punto está "marcado" por los dos tags, por tanto, por dos potencias diferentes. Se supone la situación del Ejemplo 28.

ID_PUNTO	ID_TAG	POTENCIA_MEDIA
1	06	141
1	07	126
2	06	141
2	07	121
3	06	147
3	07	133

Ejemplo 28. Datos de identificadores de punto, tag y potencia media

Como se puede observar en dicho Ejemplo 28, parece claro que, en primer lugar, tomar como referencia la potencia de una sola tag puede resultar insuficiente pues se podrían obtener puntos erróneos. Por tanto, resulta imprescindible tener en cuenta la potencia de ambas tags, tal y como se ha indicado anteriormente.

La problemática del proceso se encuentra si los puntos obtenidos son diferentes. Siguiendo el ejemplo del Ejemplo 28, se supone que se obtiene Id_Tag = 06 con potencia 141, por lo tanto se tendría el punto 1 o el punto 2 pues el valor absoluto de la resta entre los valores es cero (141 – 141). A su vez, se obtiene Id_Tag = 07 con potencia 133 por lo que se tendría el punto 3. Como se puede observar, se han obtenido tres puntos diferentes por lo que no se puede garantizar un único punto como correcto.

Para resolver el problema y obtener dicho punto se ha de comparar el par completo del punto completo con el objetivo de conocer los valores de potencia más cercanos a los obtenidos. Para ello, para los puntos 1 y 2 se ha de restar el valor 133 (obtenido con la Id_Tag = 7) a las potencias con Id_Tag = 07 de esos puntos: 133 – 126 en el caso del punto uno y 133 – 121 en el caso del punto dos. Para el punto 3 se ha de realizar el mismo proceso pero en este caso con el Id_Tag = 6, de forma que: ABS[141 – 147] = 6. Como el valor de la diferencia en este caso es menor que en el resto de puntos, se puede garantizar que el punto correcto es el 3.

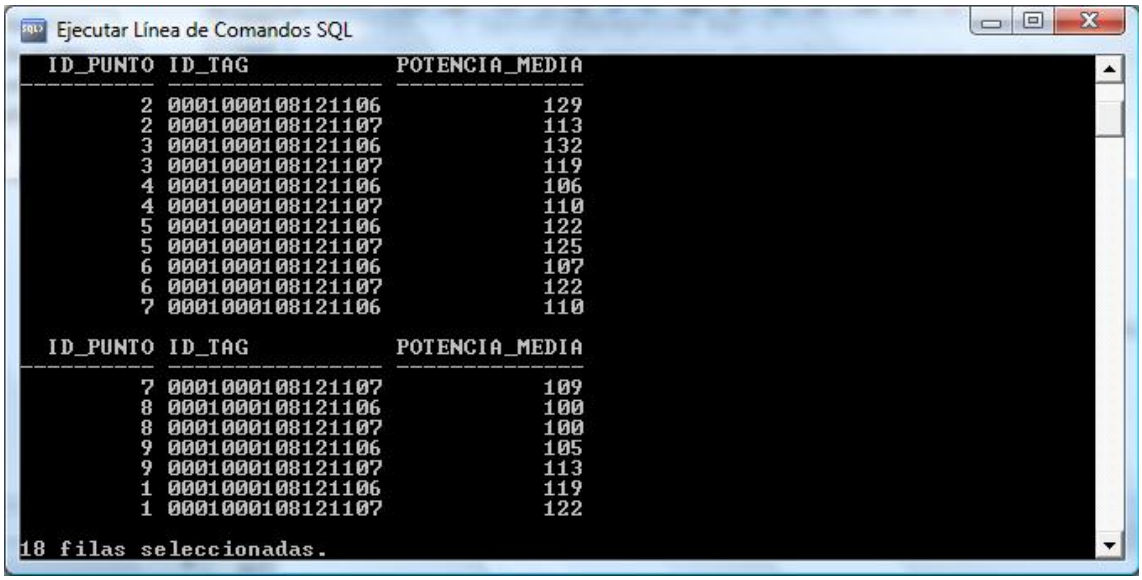
En definitiva, el proceso se basa, en caso de tener varios puntos posibles, en comparar el par completo para conseguir calcular qué potencias son más cercanas y por tanto, qué punto es el correcto.

- El sistema devuelve el punto. Una vez realizado lo anterior se devuelve el punto obtenido.

Cabe destacar, que tras las pruebas realizadas se ha observado que esta funcionalidad es válida para extraer las conclusiones pertinentes.

Antes de mostrar los resultados de las pruebas obtenidas es necesario indicar que se ha detectado un problema que ha complicado la experimentación y que está relacionado con la eficacia de la comunicación entre las etiquetas y el lector. A continuación se pasa a explicar el mismo.

Resulta muy común que desde un punto concreto del espacio se tomen medidas de potencia de manera repetida (una toma de medida después de otra) y que los resultados (potencias) obtenidos sean muy dispares sin haber introducido obstáculos de por medio y sin mover el lector ni las etiquetas entre toma y toma. Este problema incrementa la problemática de decidir que punto de los existentes es el que se debe devolver. Un ejemplo de esto se puede ver en el Ejemplo 29. En el mismo aparecen distintos puntos y la potencia de cada uno para cada tag. Se puede observar como los pares de potencia son distintos en cada punto, lo que podría llevar a pensar que los puntos están ubicados en distintos lugares, sin embargo estas medidas están hechas con el lector siempre sobre el mismo punto del espacio, sin haber movido las etiquetas, y sin haber introducido nada entre lector y etiqueta.



Ejemplo 29. Vista de línea de comandos SQL con datos de la tabla Punto_Tag

5.1. Resultados obtenidos

A continuación se describen y detallan los resultados obtenidos en cada uno de las pruebas realizadas. En primer lugar, la Tabla 11 muestra los resultados obtenidos en al primer prueba.

Medición \ Puntos	P1	P5	P9
Medición 1	1	1	1
Medición 2	1	1	1
Medición 3	0	1	1
Medición 4	1	0	1
Medición 5	1	1	1
Medición 6	1	0	1
Medición 7	1	1	1
Medición 8	1	1	1
Medición 9	1	1	1
Medición 10	1	1	1
Promedio	**0.9**	**0.8**	**1**

Tabla 11. Resultados de la prueba 1

Para esta primera prueba, al modelar el escenario con únicamente tres puntos a una distancia de 180 cm entre ellos, la precisión ha sido excelente. Los puntos se encuentran separados a una distancia bastante considerable para que poder conocer prácticamente de forma perfecta el punto donde se encuentra el usuario.

La probabilidad de acierto global es de: (0.9 + 0.8 + 1) / 3 = 0.9

En la Tabla 12 se muestran los resultados obtenidos en la segunda prueba.

Medición \ Puntos	P1	P3	P5	P7	P9
Medición 1	1	1	0	1	1
Medición 2	1	0	1	1	1
Medición 3	0	1	1	1	1
Medición 4	0	1	0	1	1
Medición 5	1	1	1	1	1
Medición 6	1	0	0	0	1
Medición 7	1	1	1	1	1
Medición 8	0	0	1	0	0
Medición 9	1	0	1	1	1
Medición 10	0	1	0	1	1
Promedio	0.6	0.6	0.7	0.8	0.9

Tabla 12. Resultados de la prueba 2

En esta segunda prueba se cuenta con 5 puntos, separados a una distancia de 90 cm. Los resultados obtenidos no son tan satisfactorios como en la primera prueba, por lo que se deduce que a menor distancia entre los puntos existe menor probabilidad de acierto.

La probabilidad de acierto global es de: (0.6 + 0.6 + 0.7 + 0.8 + 0.9) / 5 = 0.72

En la Tabla 13 se muestra los resultados de la tercera y última prueba.

Medición \ Puntos	P1	P2	P3	P4	P5	P6	P7	P8	P9
Medición 1	0	0	0	0	1	0	1	1	1
Medición 2	0	0	0	1	0	0	1	0	1
Medición 3	1	0	1	0	0	1	1	1	0
Medición 4	0	1	1	0	0	0	0	1	1
Medición 5	1	1	1	1	1	0	0	1	1
Medición 6	0	1	1	0	1	1	1	0	1
Medición 7	1	0	1	0	0	0	0	0	0
Medición 8	1	1	0	1	1	1	1	1	1
Medición 9	1	1	1	1	1	1	1	0	0
Medición 10	1	0	0	1	1	0	1	1	1
Promedio	**0.5**	**0.5**	**0.6**	**0.5**	**0.6**	**0.4**	**0.7**	**0.6**	**0.7**

Tabla 13. Resultados de la prueba 3

En esta tercera prueba se tienen un total de 9 puntos a una distancia de 45 cm entre ellos. Los resultados son concluyentes: la tasa de acierto disminuye considerablemente respecto a las anteriores pruebas.

La probabilidad de acierto global es de: (0.5 + 0.5 + 0.6 + 0.5 + 0.6 + 0.4 + 0.7 + 0.6 + 0.7) / 9 = 0.57.

En definitiva, los resultados se recogen en la Tabla 14.

Resultados	
Granularidad	Probabilidad de Acierto
45 cm	0.57
90 cm	0.72
180 cm	0.9

Tabla 14. Resultados de las pruebas realizadas

5.2. Conclusiones de la experimentación

Una vez realizadas las pruebas, los resultados de las mismas ofrecen conclusiones bastante concluyentes, tal y como se muestra en la Figura 37.

De la probabilidad de acierto en función de la precisión se puede extraer que no es aconsejable modelar el escenario con puntos separados por 45 cm, pues la tasa de acierto no es excesivamente alta.

Con una precisión de 90 cm, la probabilidad de acierto es superior, llegando a una probabilidad ideal en el caso de una precisión de 180 cm entre los puntos. En función del modelado y de los puntos necesarios se podría usar cualquiera de estas dos últimas opciones. Hay que tener en cuenta que, a mayor número de puntos en el modelado del escenario, más completo puede llegar a ser, por lo que es imprescindible encontrar un equilibrio entre los puntos del escenario y la precisión, con el fin de garantizar una buena probabilidad de acierto.

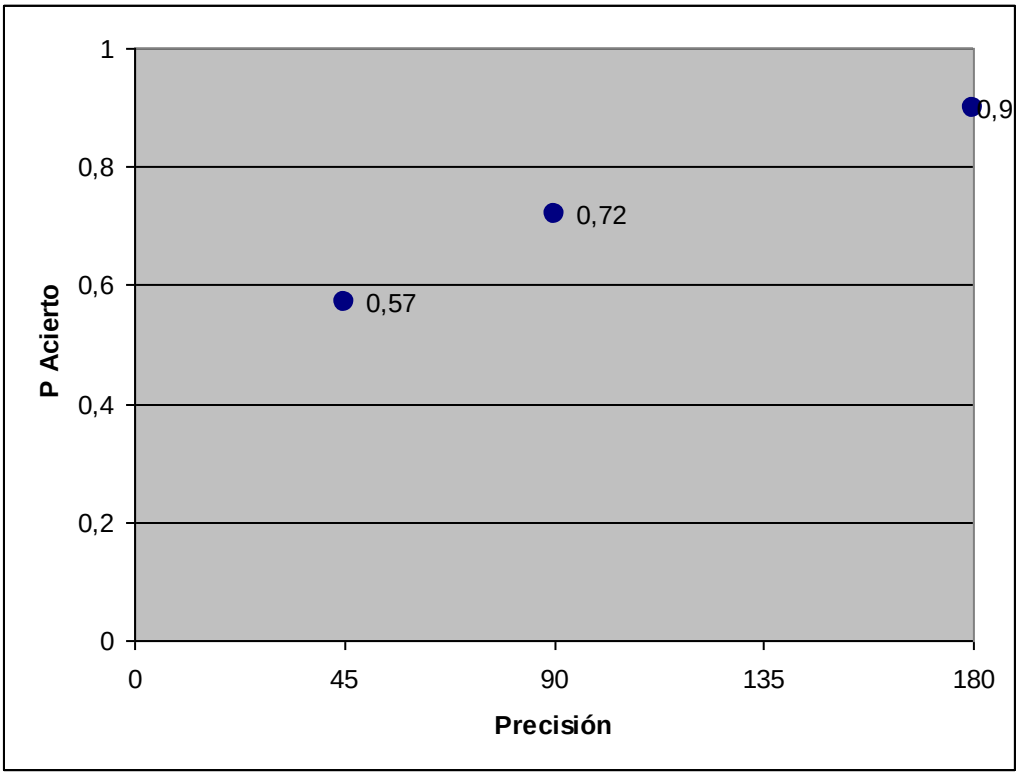


Figura 37. Gráfica de probabilidad de acierto frente a precisión

Además, utilizando el tercer experimento se pueden intentar obtener conclusiones sobre la dependencia de distancia a las tags, como se muestra en la Figura 38.

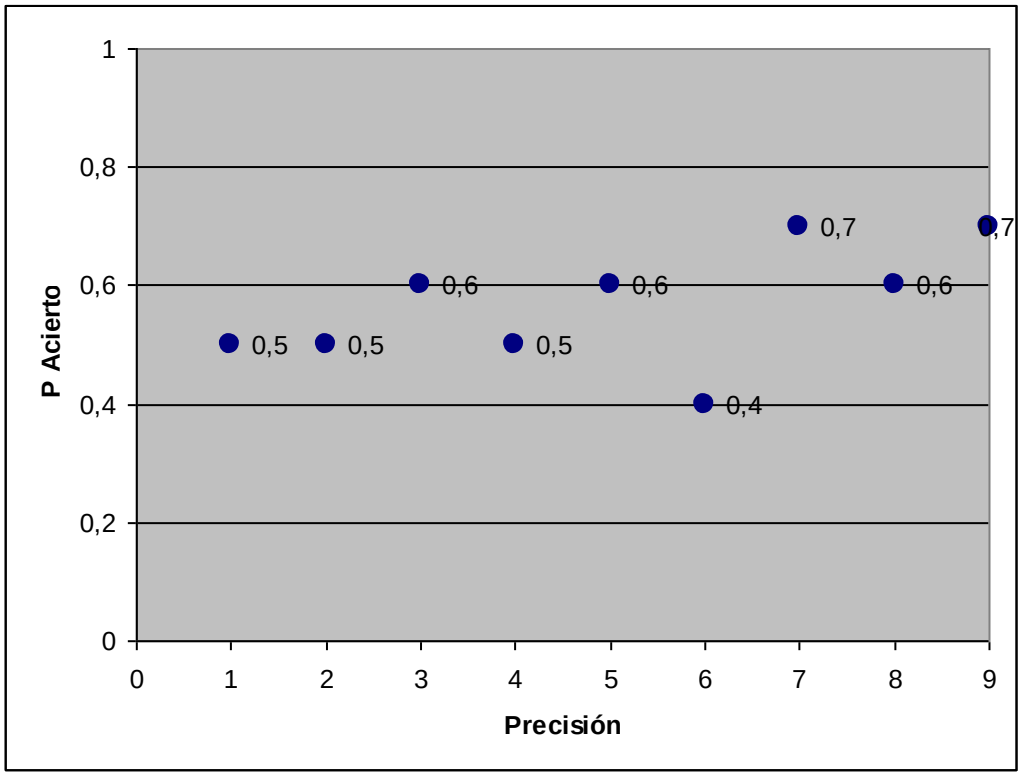


Figura 38. Gráfica de probabilidad de acierto para cada punto

Por la experimentación realizada, se ha podido observar algo de conflicto en los puntos en los que ambas tags se encontraban a similar distancia.

Conflicto entre tags. Anticolisión.

En el capítulo de Estado del Arte se pudo conocer que algunos modelos de etiquetas RFID gestionan internamente el proceso de colisión, de forma que unas esperen el turno de otras para enviar información.

Durante el proceso de pruebas se ha podido observar, tal y como se detalla también en el capítulo de Implementación, que la lectura de tags no es equitativa ni consecutiva, es decir, el lector no lee las tags de forma consecutiva, una por una, sino que se puede dar el caso de que el lector lea varias veces seguidas la misma etiqueta.

Para resolver este problema, como se ha detallado en dicho capítulo de Implementación, ha sido necesario implementar controles para seguir obteniendo tramas con el fin de leer ambas etiquetas, no únicamente una. Para líneas futuras, sobre todo en el caso de que existan más tags, es preciso estudiar esto más en profundidad.

6. Conclusiones y Líneas Futuras

Este capítulo recoge todas las conclusiones obtenidas tras el desarrollo del presente proyecto, así como las posibles líneas futuras de trabajo que podrían potenciar y mejorar el proyecto.

Cabe recordar que el proyecto tenía como objetivos principales conocer de forma teórica y práctica la tecnología RFID así como la implementación de una aplicación, utilizando dicha tecnología, para localización o posicionamiento de personas en un escenario previamente modelado. Este segundo objetivo conllevaba también la utilización y por tanto aprendizaje del uso de lenguaje programación JAVA y de bases de datos espaciales.

El primer objetivo ha sido, personalmente, muy gratificante pues ha aportado una visión e introducción a un sistema hardware en pleno crecimiento. Aunque el uso de esta tecnología hace que el sistema sea más complejo desde el punto de vista de la arquitectura e implementación del mismo, también hace que sea más completo y más enriquecedor. Además, de cara al futuro, el uso de RFID en este sistema puede servir como guía para posibles estudios y análisis.

En cuanto a la implementación se pueden obtener conclusiones positivas. La utilización de Java como lenguaje de programación y NetBeans como entorno de programación ha permitido desarrollar una aplicación estructurada en módulos que consigue obtener los datos del hardware y posibilita la interactuación del usuario con el sistema gracias a una interfaz amigable.

Gracias a estas herramientas, el sistema implementado permite identificar la localización o posición de un nodo dentro de un escenario modelado previamente por un administrador del sistema, incluyendo una interfaz en la que se representa la información obtenida por el sistema.
A su vez, la utilización de una base de datos espacial ha permitido, además de almacenar los datos, poder representarlos posteriormente como datos espaciales o geométricos gracias a las características y funciones del paquete Oracle Spatial.

No obstante, a pesar de estas conclusiones positivas, el proyecto puede sentar las bases de un estudio futuro, tanto de diseño como de la implementación, que logren un sistema

más completo. Se presentan, por tanto, algunas de las líneas futuras que pueden tenerse en cuenta para posibles análisis posteriores:

1. Mejora tecnológica: como se ha indicado a lo largo del presente documento, la tecnología RFID se encuentra en pleno proceso de crecimiento, teniendo en el mercado modelos de lectores y etiquetas cada vez mejores (con mayor alcance, más eficaces, cómodos y en definitiva más completos). Ante la problemática del uso de un lector RFID con conexión al puerto de serie RS-232, podría resultar interesante el uso de un lector RFID con un interfaz más moderno, así como unas etiquetas con mayor alcance que incluso puedan recibir y emitir señales a pesar de la existencia de obstáculos.

2. Añadir más etiquetas al proyecto: sería muy interesante poder trabajar con un número mayor de etiquetas RFID, con el objetivo de utilizar el sistema en escenarios más amplios y completos. Esta situación, además, potenciaría la eficacia y utilidad del sistema.

3. Nuevos factores en la toma de medidas: durante la experimentación se ha observado que las medidas no resultaban del todo óptimas en el caso de encontrar obstáculos en el escenario. Esta situación debe ser análoga en el caso de la altura de las etiquetas, la orientación de las mismas, etc., por lo que, unida a la primera línea futura detallada anteriormente, podría resultar muy interesante controlar y analizar este tipo de parámetros o restricciones.

4. Personalización del escenario: la interfaz realizada, en lo que a la creación de la red se refiere, es un tanto rígida, por lo que la implementación de opciones de personalización dotarían al sistema de una amigabilidad y utilidad más interesante para el usuario. Algunos ejemplos de cosas que sería interesante añadir son: utilización de un escenario con mayor o menor número de casillas, distintas formas del mismo (rectangular por ejemplo), modificación de la escala del escenario, o añadir una imagen (ImageIcon) en lugar de representar una cuadrícula.

5. Mayor número de escenarios: se ha implementado el sistema con un solo escenario. La posibilidad de utilizar varios escenarios completaría el uso y alternativas para el usuario. En principio, por las valoraciones realizadas, dicha implementación no conllevaría demasiados cambios. Principalmente, implicaría la creación de una nueva tabla en la base de datos que albergara cada escenario, y algunos cambios en la interfaz para que permita elegir entre los diversos escenarios.

6. Eventos de ratón en el escenario: puede tenerse en cuenta dentro de la línea futura número 4, no obstante, se ha puesto aparte porque en este caso, durante la realización del proyecto, se realizó un análisis sobre el uso de eventos de ratón para señalar casillas en el escenario, y en el código de la aplicación ha quedado plasmada parte de lo que podría ser esa funcionalidad.

7. Gestión de usuarios: en la actualidad la creación de usuarios se realiza directamente en la base de datos, por medio de scripts ejecutados por el administrador del sistema. Podría resultar interesante realizar dicha gestión dentro de la aplicación, analizando e incluso ampliando el número de perfiles actuales (administrador y consultor).

8. Sentencias preparadas (PreparedStatement): por falta de tiempo no se ha podido realizar esta funcionalidad que dotaría de mayor seguridad al sistema, ya que las sentencias preparadas evitarían en mayor medida la inyección SQL (SQL Injection). Normalmente, estas funcionalidades se realizan en servicios web que pueden ser susceptibles de ataques externos.
   Por el contrario, en cuanto a la autentificación de usuarios se refiere, se ha implementado la posibilidad de encriptar la contraseña de los mismos.

9. Estudio pormenorizado de la clase java Paint: la representación de componentes visuales en Java ha sido uno de los aspectos más complejos del proyecto. Un estudio amplio y pormenorizado de la clase Paint, así como de los métodos paint(), repaint(), y update(), podría refinar la representación de componentes.

10. Control de campos de texto y evitar duplicidades: en cuanto al código de implementación se refiere, sería preciso realizar un control interno de los datos que el usuario introduce en los distintos campos de texto (TextField) de los que consta la aplicación para evitar datos repetidos. Esto podría evitar, entre otros aspectos, que se creen elementos varias veces, como por ejemplo, la creación de varios enlaces que unan los mismos nodos.

Acrónimos

A continuación se definen los acrónimos utilizados a lo largo del documento con el objetivo de aclarar su uso y significado.

- **BLN**: Bluetooth Location Network. Sistema de localización basado en la búsqueda de dispositivos bluetooth.
- **COM**: también llamado puerto de serie o serial. Es una interfaz de comunicaciones de datos digitales, frecuentemente utilizada en ordenadores y periféricos.
- **EPC**: Electronic Product Code o Código Electrónico de Producto. Es un número uníco diseñado para identificar de forma unívoca cualquier objeto.
- **dBm**: es una unidad de medida utilizada para expresar la potencia absoluta mediante una relación logarítmica.
- **GB**: es la abreviatura de Gigabyte, que es una unidad de almacenamiento de información que equivale a 10^9 bytes
- **GIS**: Sistema de Información Geográfica (Geographic Information System en inglés). Unifica hardware, software y datos geográficos que gestiona la información geográfica que se obtiene gracias a estas herramientas,
- **GPS**: Sistema de Posicionamiento Global (Global Positioning System en inglés). Es un sistema global de navegación por satélite.
- **IDE**: abreviatura anglosajona de Integrated Development Environment, es decir, es un software o entorno de trabajo utilizado para que los programadores desarrollen aplicaciones.
- **IEEE**: Institute os Electrical and Electronics Engineers o Instituto de Ingenieros Eléctricos y Electrónicos. Es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.
- **ISO**: Organización Internacional para la Estandarización. Es un organismo encargado de promover el desarrollo y cumplimiento de normas internacionales de fabricación, comercio y comunicación.

- **mW**: es un submúltiplo de la unidad de watio, en concreto, mW significa milivatio.
- **NAVSTAR**: análogo al GPS. Es la abreviatura inglesa de Navigation Satellite Timing And Ranging.
- **OMG**: Grupo de Gestión de Objetos. Es un consorcio que se dedica al establecimiento de estándares de tecnologías orientadas a objetos.
- **ONS**: Object Naming Service. Es un mecanismo para descubrir información de un producto y los servicios relacionados con el Código Electrónico de Producto (EPC).
- **PDA**: del inglés Personal Digital Assistant, es una dispositivo de mano que puede realizar muchas funciones, aunque originariamente se diseñó como agenda electrónica.
- **RAM**: memoria de acceso aleatorio (Random-Access Memory en inglés), es de la memoria desde donde el procesador recibe las instrucciones y guarda los resultados.
- **RFID**: siglas de Radio Frequency Identification, en español Identificación por radiofrecuencia. Es un sistema que permite la comunicación entre dispositivos como las etiquetas y lectores RFID.
- **RSSI**: es la abreviatura en inglés de Receive Signal Strength Indication, es decir, Indicador de fuerza de señal de recepción. Indica el nivel de fuerza de las señales recibidas en los dispositivos RFID, entre otros.
- **SGBD**: Sistema de Gestión de Bases de Datos. Son un tipo de software que hace de intermediario entre la base de datos, el usuario y las aplicaciones que la utilizan.
- **SQL**: Structured Query Language, en español Lenguaje de consulta estructurado. Es un lenguaje declarativo de acceso y gestión de bases de datos.
- **TCP/IP**: referencia los dos protocolos de comunicación más importantes: TCP (Protocolo de Control de Transmision) e IP (Protocolo de Internet).
- **UHF**: siglas del inglés Ultra High Frequency, es decir, Frecuencia Ultraalta. Es una banda del espectro electromagnético que ocupa el rango de frecuencias de 300 MHz a 3 GHz.

- **UML**: Unified Modeling Language, en español Lenguaje Unificado de Modelado. Es el lenguaje de modelado de sistemas software más utilizado.
- **USB**: Bus universal en serie. Es un puerto que sirve para conectar periféricos al ordenador.
- **WiFi**: es una tecnología que permite la utilización de redes inalámbricas. Generalmente se utiliza para acceder a Internet.

Bibliografía y Referencias

En este apartado se detalla la bibliografía y referencias utilizadas a lo largo del documento.

[41] **Adoración de Miguel, Paloma Martínez, Elena Castro, José Mª Cavero, Dolores Cuadra, Ana Mª Iglesias y Carlos Nieto. Diseño de Bases de Datos. Problemas resueltos**. Año 2001. Editorial RA-MA. 1º Edición. 482 páginas.

[40] **Adoración de Miguel y Mario Piattini. Fundamentos y modelos de Bases de Datos**. Año 1999. Editorial RA-MA. 2ª Edición. 498 páginas.

[1] **AETIC. La tecnología RFID: usos y oportunidades.** [http://www.aetic.es/CLI_AETIC/ftpportalweb/documentos/RFIDCOMPLETO.pdf] Fecha de revisión: 17 de septiembre de 2010.

[4] **AIP Systems. Soluciones de posicionamiento. Tecnologías aplicables**. [http://www.aipsystems.com/a_trazatecnologia.html] Fecha de revisión: 17 de septiembre de 2010.

[38] **Carme Martín Escofet – UOC. El lenguaje SQL**. [http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02149.pdf]. Fecha de revisión: 17 de septiembre de 2010.

[18] **David Alayón - Techtear.com. RFID, el nuevo código de barras**.

[http://www.techtear.com/2007/02/27/rfid-el-nuevo-codigo-de-barras/]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 27 de febrero de 2007.

[51] **Descarga de NetBeans**. [http://netbeans.org/downloads/index.html]. Fecha de revisión: 17 de septiembre de 2010.

[7] **Domingo Morales L. y Javier Ruiz del Solar. Sistema biométricos: Reconocimiento de huellas dactilares mediante transformada de Hough generalizada**. [http://www2.ing.puc.cl/~iing/ed429/sistemas_biometricos.htm]. Fecha de revisión: 17 de septiembre de 2010.

[16] **Ekahau. Ekahau HeatMapper**.

[http://www.ekahau.com/products/heatmapper/features-a-screen-shots.html]. Fecha de revisión: 17 de septiembre de 2010.

[17] **Electrónica-basica.com. Algo de historia RFID**. [http://www.electronica-basica.com/historia-rfid.html]. Fecha de revisión: 17 de septiembre de 2010.

[26] **EPC global. CETECOM lanza el primer sistema a nivel mundial para la evaluación de confirmidad de equipos de RFID EPC GEN2**. [http://www.epcglobalsp.org/EPCnoticias/Cetecom]. Fecha de revisión: 17 de septiembre de 2010.

[27] **EPC global. Object Naming Service (ONS) Version 1.0**. [http://www.epcglobalinc.org/standards/ons/ons_1_0-standard-20051004.pdf]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 4 de octubre de 2005.

[28] **EPC global. Object Naming Service (ONS) Version 1.0.1.** [http://www.epcglobalinc.org/standards/ons/ons_1_0_1-standard-20080529.pdf]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 29 de mayo de 2008.

[32] **Franco Canestri. Agilent Technologies Gmbh & Co. RFID: Descripción general sobre su aplicación y retos a la hora de realizar las pruebas**.
[http://www.redeweb.com/_txt/636/86.pdf]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: noviembre de 2007.

[46] **Giovynet driver**. [http://java.giovynet.com/Giovynet/]. Fecha de revisión: 17 de septiembre de 2010.

[25] **GS1 Ecuador. EPC**.
[http://www.gs1ec.org/contenido/index.php?option=com_content&view=article&id=60&Itemid=78#4]. Fecha de revisión: 17 de septiembre de 2010.

[13] **GSMSpain. Definición de WIFI**.

[http://www.gsmspain.com/glosario/?palabra=WIFI]. Fecha de revisión: 17 de septiembre de 2010.

[30] **IGT Microelectronics. Tecnología RFID: aplicaciones**.

[http://www.igt.es/RFID/]. Fecha de revisión: 17 de septiembre de 2010.

[20] **ISO/IEC 10536-1. Identification Cards - Contactless integrated circuit(s) cards – Closed coupled cards**. [http://webstore.iec.ch/preview/info_isoiec10536-1%7Bed2.0%7Den.pdf]. Fecha de revisión: 17 de septiembre de 2010.

[12] **Jaime José García Reinoso. Contribución al desarrollo de aplicaciones alternativas de Bluetooth: localización de usuarios y telemando**. Abril de 2003. Leganés, Madrid. Tesis Doctoral.

[50] **Java Cryptography Extension (JCE), Reference Guide**.

[http://download.oracle.com/javase/1.4.2/docs/guide/security/jce/JCERefGuide.html]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 10 de enero de 2002.

[54] **Java. Descargar Java para Windows**.

[http://www.java.com/es/download/chrome.jsp?locale=es&host=www.java.com]. Fecha de revisión: 17 de septiembre de 2010.

[19] **Jorge Alberto Alvarado Sánchez. Sistema de control de acceso con RFID**. Enero de 2008. Méjico, D.F. Tesis Doctoral.

[5] **José Antonio E. García Álvarez. Así funciona el GPS. GPS diferencial**. [http://www.asifunciona.com/electronica/af_gps/af_gps_13.htm] Fecha de revisión: 17 de septiembre de 2010.

[39] **José Enrique González Cornejo. ¿Qué es UML?** [http://www.docirs.cl/uml.htm]. de revisión: 17 de septiembre de 2010.

[44] **Kimaldi. Lector RFID Activo SYRD 245-2**.

[http://www.kimaldi.com/productos/sistemas_rfid/lectores_rfid_y_tags_activos/lectores_de_rfid_activos/lector_rfid_activo_syrd245_2]. Fecha de revisión: 17 de septiembre de 2010.

[43] **Kimaldi. Tag RFID Activo SYTAG245-2K**.

[http://www.kimaldi.com/productos/sistemas_rfid/lectores_rfid_y_tags_activos/tags_rfid_activos/tag_rfid_activo_sytag245_2k]. Fecha de revisión: 17 de septiembre de 2010.

[34] **Knut Stolze. SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems**.

[http://doesen0.informatik.uni-leipzig.de/proceedings/paper/68.pdf]. Fecha de revisión: 17 de septiembre de 2010.

[31] **Mecalux logismarket. Información detallada del producto Sistema RFID de identificación de ganado**. [http://www.logismarket.com.mx/devbus/sistema-rfid-de-identificacion-de-ganado/1322572061-1179567384-p.html]. Fecha de revisión: 17 de septiembre de 2010.

[15] **Miguel Ángel Criado - Público.es. Los autobuses de Madrid tendrán WiFi gratuito**.[http://www.publico.es/ciencias/336926/autobuses/madrid/wifi/gratuito]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 16 de septiembre de 2010.

[49] **Oracle Java SE Documentation. Java ™ Cryptography Architecture (JCA) Reference Guide: The Cipher Class**.

[http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html#Cipher]. Fecha de revisión: 17 de septiembre de 2010.

[48] **Oracle Java SE Documentation. Java ™ Cryptography Architecture (JCA) Reference Guide: The KeyGenerator Class**.

[http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html#KeyGenerator]. Fecha de revisión: 17 de septiembre de 2010.

[52] **Oracle España. Oracle Database 10g Express Edition**.

[http://www.oracle.com/lang/es/database/Express_Edition.html]. Fecha de revisión: 17 de septiembre de 2010.

[56] **Oracle. Java SE Downloads: JDK 6 Update 21**.

[http://www.oracle.com/technetwork/java/javase/downloads/jdk6-jsp-136632.html]. Fecha de revisión: 17 de septiembre de 2010.

[35] **Oracle® Spatial. User's Guide and Reference**. Fecha de publicación: marzo de 2006.

[37] **Página web de Eclipse**. [http://www.eclipse.org/]. Fecha de revisión: 17 de septiembre de 2010.

[36] **Página web de NetBeans**. [http://netbeans.org/] Fecha de revisión: 17 de septiembre de 2010.

[47] **Página web de SYRIS Xtive RFID System**. [http://xtive.com.tw/]. Fecha de revisión: 17 de septiembre de 2010.

[10] **Página web Oficial de Bluetooth**.

[http://www.bluetooth.com/Spanish/Technology/Pages/default.aspx]. Fecha de revisión: 17 de septiembre de 2010.

[29] **RFID Magazine. Presentan el primer carro de "compra" inteligente**. [http://www.rfid-magazine.com/noticias/detalle.php?id=1479]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 16 de diciembre de 2008.

[42] **RFID point. Tags Activos, Pasivos y Semi Pasivos**.

[http://www.rfidpoint.com/preguntas-frecuentes/tags-activos-pasivos-y-semi-pasivos/]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 15 de julio de 2009.

[21] **Standing Document 1: WG8 Projects**. [http://wg8.de/sd1.html]. Fecha de revisión: 17 de septiembre de 2010. Fecha de publicación: 26 de noviembre de 2009.

[9] **Tecno Soft. Simbologías de códigos de barras**. [http://www.tecno-symbol.com/simbologias.html]. Fecha de revisión: 17 de septiembre de 2010.

[45] **Tropic Hardware y Electrónica. Estándar de comunicaciones RS232**. [http://www.euskalnet.net/shizuka/rs232.htm]. Fecha de revisión: 17 de septiembre de 2010.

[11] **Wikipedia. Bluetooth**. [http://es.wikipedia.org/wiki/Bluetooth]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 17 de septiembre de 2010.

[8] **Wikipedia. Código de barras**. [http://es.wikipedia.org/wiki/Código_de_barras]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 17 de septiembre de 2010.

[2] **Wkipedia. dBm.** [http://es.wikipedia.org/wiki/DBm] Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 31 de agosto de 2010.

[14] **Wikipedia. Estándar IEEE 802.11**. [http://es.wikipedia.org/wiki/IEEE_802.11]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 17 de septiembre de 2010.

[6] **Wikipedia. GPS**. [http://es.wikipedia.org/wiki/GPS] Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 17 de septiembre de 2010.

[22] **Wikipedia. ISO/IEC 14443**. [http://en.wikipedia.org/wiki/ISO/IEC_14443]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 11 de septiembre de 2010.

[23] **Wikipedia. ISO/IEC 15693**. [http://en.wikipedia.org/wiki/ISO/IEC_15693]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 3 de julio de 2010.

[24] **Wikipedia. ISO/IEC 18000**. [http://en.wikipedia.org/wiki/ISO/IEC_18000]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 2 de julio de 2010.

[57] **Wikipedia. Java Archive**. [http://es.wikipedia.org/wiki/Java_Archive]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 30 de mayo de 2010.

[55] **Wikipedia. Java Development Kit**.

[http://es.wikipedia.org/wiki/Java_Development_Kit]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 3 de junio de 2010.

[53] **Wikipedia. Máquina virtual Java**.

[http://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 21 de junio de 2010.

[33] **Wikipedia. Open Geospatial Consortium**.

[http://es.wikipedia.org/wiki/Open_Geospatial_Consortium]. Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 25 de junio de 2010.

[3] **Wikipedia. Triangulación.** [http://es.wikipedia.org/wiki/Triangulación] Fecha de revisión: 17 de septiembre de 2010. Fecha de última modificación: 12 de agosto de 2010.

Anexo I. Planificación

Todo desarrollo software tiene inherente un ciclo de vida que abarca desde su fase inicial hasta su fase final. Las tareas que se fijan en este ciclo de vida del desarrollo software marcan los hitos que se han de cumplir, y por ello, resulta imprescindible planificar y dividir dichas tareas.

Para este Proyecto se ha contado con un único recurso humano por lo que se ha procedido a dividir el desarrollo software en 4 etapas claramente diferenciadas, en el que al final de cada una de ellas se reúne y elabora la documentación e información necesarias y se realiza la verificación y validación pertinente antes de pasar a la siguiente etapa. Además, cabe destacar que en cada una de las etapas se incluyen las tareas pertinentes relacionadas con la elaboración de la documentación correspondiente, que será presentada finalmente cuando se acepte el proyecto. Dichas etapas son las siguientes:

- Etapa de Planificación y Definición de Requisitos: es la etapa de concepción del proyecto. Durante ésta se estudia el modelo de negocio, se toman y definen los requisitos necesarios, y por tanto, se estudian los objetivos que el desarrollo software debe cumplir. Es una etapa de especial relevancia e importancia ya que el resultado final del proyecto estará muy ligado a la información recogida, analizada y dispuesta durante esta etapa.

- Etapa de Análisis y Diseño: una vez tomados los requisitos necesarios y recogida la información necesaria se procede a analizarla en profundidad. Al final de esta etapa la documentación de requisitos será completa y fiable. Durante esta etapa se estudian los recursos materiales necesarios y se procede a realizar el diseño del sistema que posteriormente será desarrollado.

- Etapa de Implementación: como su nombre indica en esta etapa se realiza el desarrollo del sistema en función de la información extraída y analizada en las etapas previas. Al final de esta etapa el sistema quedará desarrollado

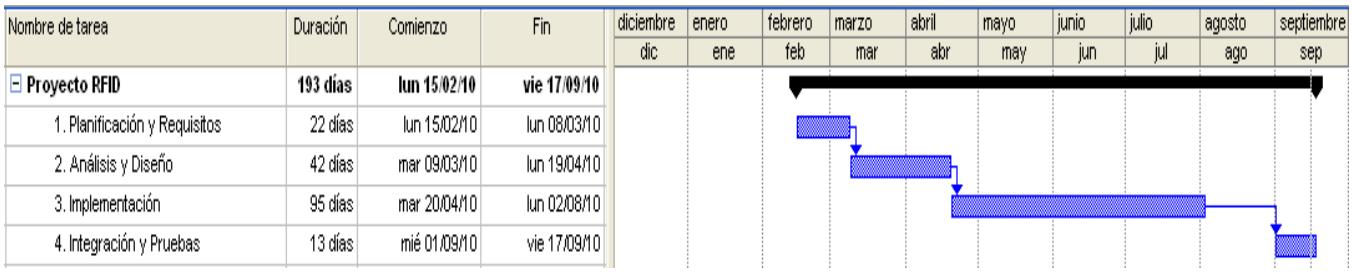
completamente a falta de un testeo en profundidad. Además, si procede, se elaborará un manual de usuario.

- Etapa de Integración y Pruebas: una vez elaborado el sistema, aunque previamente ya se han debido realizar pruebas del mismo, será en esta etapa en la que se realiza un testeo en profundidad, es decir, se realizan las pruebas de aceptación del proyecto, tanto unitarias como de integración de los subsistemas. Para ello, se debe elaborar un documento que recoja qué pruebas deben realizarse y su resultado. Normalmente, una vez validadas dichas pruebas, se procede a instalar y configurar el desarrollo software en las máquinas correspondientes, es decir, se pone el desarrollo en Producción para que el cliente comience a utilizarlo.

  Por último, se está en disposición de sellar la aceptación del proyecto.

Para plasmar de forma más visual la duración de las etapas en las que se ha dividido el proyecto, se realiza un diagrama de Gantt, como el que se muestra en la Figura 39.

Se toma como fecha de comienzo del Proyecto el día 15 de febrero de 2010, fecha en la que se comenzó a identificar las necesidades e ideas sobre el mismo y como fecha de fin el día 17 de septiembre de 2010. Cabe destacar que no se ha tomado un calendario laboral estándar, de lunes a viernes, sino un calendario que incluye trabajo los fines de semana pues el autor del proyecto ha dedicado tiempo al proyecto durante estos periodos.



Nombre de tarea	Duración	Comienzo	Fin
Proyecto RFID	**193 días**	**lun 15/02/10**	**vie 17/09/10**
1. Planificación y Requisitos	22 días	lun 15/02/10	lun 08/03/10
2. Análisis y Diseño	42 días	mar 09/03/10	lun 19/04/10
3. Implementación	95 días	mar 20/04/10	lun 02/08/10
4. Integración y Pruebas	13 días	mié 01/09/10	vie 17/09/10

Figura 39. Calendario de Gantt con la planificación del proyecto

Como se puede observar en la figura 39, la duración total del proyecto ha sido de 193 días, desglosado en las 4 etapas citadas anteriormente, y contando el mes de agosto como periodo estival. La duración de cada etapa es la siguiente: la etapa de Planificación y Requisitos tiene una duración de 22 días, la de Análisis y Diseño una duración de 42 días, la etapa de Implementación una duración de 95 días, y por último, la etapa de Integración y Pruebas una duración de 13 días. Se ha de tener en cuenta que durante el transcurso del proyecto, en ocasiones, se ha precisado revisar la planificación y diseño del proyecto por lo que en algún momento de una etapa se ha realizado trabajo de la anterior. Un ejemplo ha sido la etapa de implementación, que ha precisado algunas revisiones de las etapas anteriores, como en el caso del uso del convertidor USB para realizar la conexión del lector RFID con el equipo. Por otro lado, se ha de tener en cuenta que cada fase incluye la elaboración y revisión de la documentación pertinente, que ha permitido optimizar el tiempo de elaboración del presente documento.

Anexo II. Presupuesto

Un proyecto de desarrollo software debe tener un presupuesto que sustente y exponga, en términos económicos, el trabajo realizado.

Para elaborar el presupuesto se han de tener en cuenta todas las particularidades que sean susceptibles de otorgar valor al proyecto por lo que, normalmente, el presupuesto se divide en costes humanos y costes de material.

Los **costes de recursos humanos** describen los costes que conlleva el trabajo realizado por las personas participantes en el proyecto, desde el Jefe de Proyecto hasta el analista o el programador, entre otros. En este caso, sólo se cuenta con un recurso humano, que ha sido el encargado de llevar a buen puerto el proyecto, por lo que posteriormente se describen los gastos para este caso. Cabe destacar que se ha tomado como calendario laboral de lunes a domingo, teniendo en cuenta que cada jornada laboral consta de 4 horas diarias de trabajo.

Los **costes de recursos materiales** describen los costes relativos al gasto de material necesario para realizar el proyecto. Se deben incluir en este análisis los gastos ocasionados por la adquisición de equipos, dispositivos hardware e incluso licencias de software. Si procede, se pueden incluir también gastos varios, tales como gasto de luz, transporte o alquiler del local.

Costes de recursos humanos

En lo referente a los costes de personal que ha participado en el proyecto, se ha tenido en cuenta que ha sido una única persona la encargada de realizarlo, y que tiene un coste de 15 euros por hora por sus funciones de analista-programador.

Como se ha descrito en el anexo de Planificación, han sido 4 las etapas que comprenden el proyecto. En la Tabla 15 se detalla el número de horas de cada una de ellas y los costes humanos correspondientes.

Etapa	Descripción	Horas (h)	Costes Humanos (euros)
Planificación y Requisitos	Estudio de necesidades, ideas y objetivos. Así como extracción de requisitos.	88	88*15 = 1.320
Análisis y Diseño	Análisis del sistema, de los requisitos y de las necesidades. Diseño completo del sistema.	168	2.520
Implementación	Desarrollo del sistema diseñado.	380	5.700
Integración y Pruebas	Integración de los módulos software. Pruebas unitarias y pruebas de aceptación de software.	52	780
TOTAL		**688**	**10.320**

Tabla 15. Costes de personal

Por tanto, los costes de recursos humanos tienen un total de 10.320 euros.

Costes de recursos materiales

Como se indicó anteriormente, estos costes están relacionados con la adquisición de equipos, dispositivos hardware o licencias software, entre otros.

Para este proyecto se cuenta con hardware y software, los cuales se analizan a continuación.

En cuanto al hardware se precisa un portátil, junto a sus periféricos correspondientes, que tendrán una vida útil de 5 años e incluso superior para este tipo de proyecto. Por otro lado, se usan dispositivos RFID, es decir, etiquetas (tags), y un lector. Estos dispositivos tienen su origen en un proyecto anterior, por lo que queda constatado que pueden usarse en futuros proyectos de investigación. Además, se precisa un convertidor de puerto USB para poder utilizar el lector RFID con dicha conexión.

En cuanto al software, se ha optado por reducir costes para compensar el elevado coste del hardware, utilizando aplicaciones de distribución libre.

Para la implementación del sistema y creación gráfica de las pantallas, se ha utilizado el lenguaje de programación Java, sobre la aplicación NetBeans 6.8 que se puede descargar de forma gratuita en su página web oficial [51].

Por otro lado, en cuanto a base de datos, se ha utilizado Oracle 10G Express bajo una licencia de desarrollo de libre distribución que se puede descargar también desde su página web oficial [52].

Por último, para la elaboración de la documentación se ha utilizado una licencia académica de Microsoft Office.

Cabe destacar que el coste presupuestado para hardware debe tenerse en cuenta como una estimación. Ésta se muestra en la Tabla 16.

Concepto	Importe (euros)
Equipo informático portátil	400
2 Etiquetas electrónicas (Tags) RFID	60
Lector RFID	600
Convertidor de puerto	25
Licencia de Microsoft Office	180
Licencia NetBeans	Distribución libre, sin cargo.
Licencia Oracle 10G para desarrollo	Distribución libre, sin cargo.
TOTAL	**1.265**

Tabla 16. Coste de recursos materiales

Como se puede observar los costes de hardware son ciertamente elevados respecto al software. No obstante, se han tomado los costes de compra iniciales de los elementos hardware, los cuales podrán ser amortizados en sucesivos proyectos, tal y como se ha indicado previamente.

En definitiva, el resumen de los gastos totales del proyecto se reflejan en la Tabla 17.

Concepto	Importe (euros)
Costes de recursos humanos	10.320
Costes de recursos materiales	1.265
Base imponible	11.585
I.V.A. (16%)	1.853,60
TOTAL	**13.438,60 euros**

Tabla 17. Presupuesto total

Por tanto, el **presupuesto total** del proyecto tiene un importe de **13.438, 60 euros**.

Anexo III. Manual de usuario

En este anexo se detallan de forma pormenorizada todas las funcionalidades que el usuario puede utilizar en esta aplicación. Además, se detalla qué programas son necesarios para su ejecución, así como el proceso de instalación de las herramientas del entorno de trabajo.

Proceso de instalación

En primer lugar, para poder ejecutar correctamente la aplicación, es necesario tener instalado en el equipo (local) la máquina virtual de Java [53] [54], también llamado JRE, con una de las últimas versiones del JDK (Java Development Kit) [55] [56] en el caso de precisar la instalación del entorno de trabajo, como se verá más adelante.

Para implementar la aplicación se ha utilizado una base de datos instalada en local, por lo que se precisa también instalar una base de datos Oracle, aconsejando la versión 10g Oracle Express Edition [52], ya que además de ser gratuita, no consume tantos recursos como las versiones más completas.

Una vez instalada la base de datos, habría que ejecutar, por medio de línea de comandos por ejemplo, los scripts necesarios y facilitados para crear las tablas pertinentes. Normalmente este proceso es llevado a cabo por un administrador del sistema.

Instalando el entorno de trabajo.

Si se desea instalar el entorno de trabajo con el que poder consultar, ejecutar o modificar el código de la aplicación se precisa la instalación de la herramienta IDE NetBeans en su última versión (en la actualidad es la versión 6.9.1) [51].

Una vez instalada esta plataforma se debe crear un proyecto nuevo referenciando la carpeta en la que se encuentra el código de la aplicación facilitado. Esta carpeta debe contener las librerías jar [57] correspondientes, especialmente dos de ellas: la librería SerialPort con la que se podrán obtener los datos del lector RFID y la librería JDBC de Oracle, que resulta imprescindible para realizar conexiones y operaciones con la base de datos instalada en local.

Una vez configurados estos parámetros queda un último paso: la instalación de los drivers para poder utilizar el convertidor de puerto USB que virtualiza un puerto COM con el que no será necesario contar con un puerto RS-232 en el equipo. Se conecta el lector al cable convertidor y éste último a cualquier puerto USB del equipo. Los drivers vienen incluidos en el CD de instalación que acompaña al cable convertidor. Una vez terminado el proceso de instalación se debe consultar en el Administrador de Dispositivos del Sistema Operativo el número de puerto que han virtualizado los drivers. En nuestro caso, queda establecido el puerto COM4.

Utilizando la aplicación de posicionamiento

En el siguiente apartado se explica la forma de uso y las distintas funcionalidades que el usuario puede utilizar en la aplicación.

Una vez ejecutada la aplicación se presenta una ventana de inicio como la de la Figura 40, con una pestaña llamada Archivo. Si se pulsa en esa ventana se observan dos opciones: Conectar y Salir.

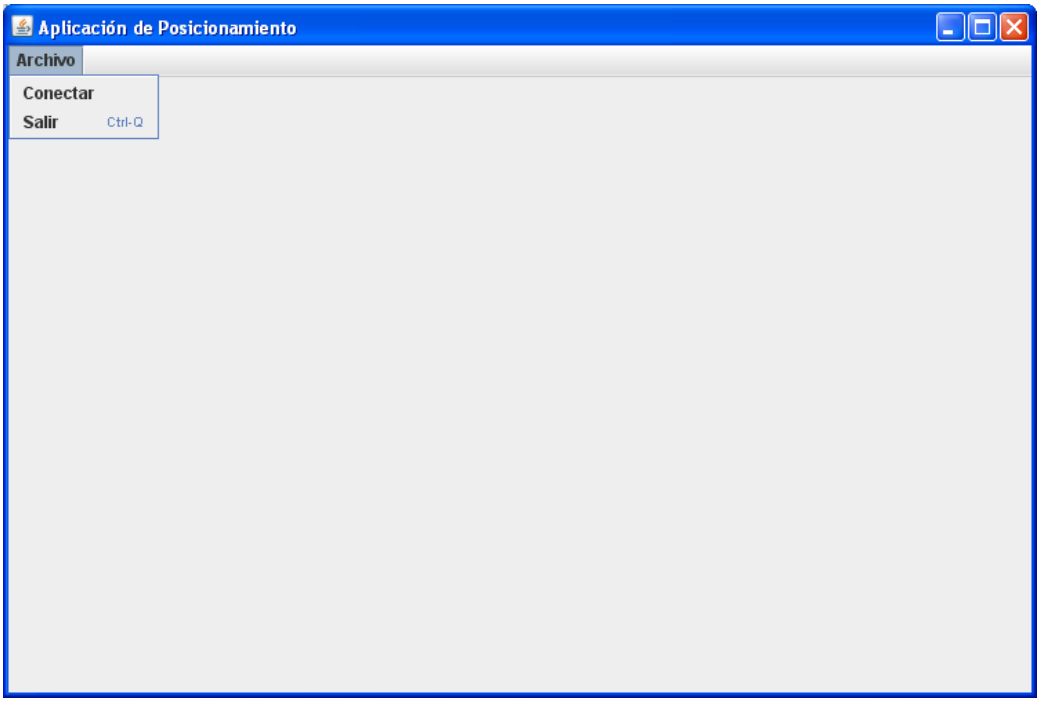


Figura 40. Ventana inicial de la aplicación

En el caso de pulsar la opción Conectar se presenta una ventana de conexión en la que el usuario debe introducir el nombre de usuario y la contraseña y pulsar el botón Aceptar, con el fin de validar sus datos, como en la Figura 41. Por otro lado, la opción Salir, cierra la aplicación.

Figura 41. Ventana de acceso a la aplicación

Una vez validados los datos de usuario, existen dos opciones posibles: que el usuario tenga un perfil Administrador o un perfil Consultor, teniendo en cuenta que los datos introducidos son correctos. En cualquiera de ambos casos aparece una nueva ventana en la que aparece un mensaje informando del perfil con el que se ha realizado la conexión y una nueva barra de menú, en la que se pueden desplegar las opciones correspondientes, según el perfil del usuario:

- En el caso de que el usuario se haya autentificado como perfil Administrador, se presenta una nueva ventana con una cuadrícula con casillas en blanco. A su vez, en la barra de menú, se puede observar una pestaña "Archivo" en la que, al pulsarla, se despliegan las distintas funcionalidades que puede realizar este tipo de usuario: Crear Punto, Eliminar Punto y también la opción de Desconectar, tal y como se muestra en la Figura 42.

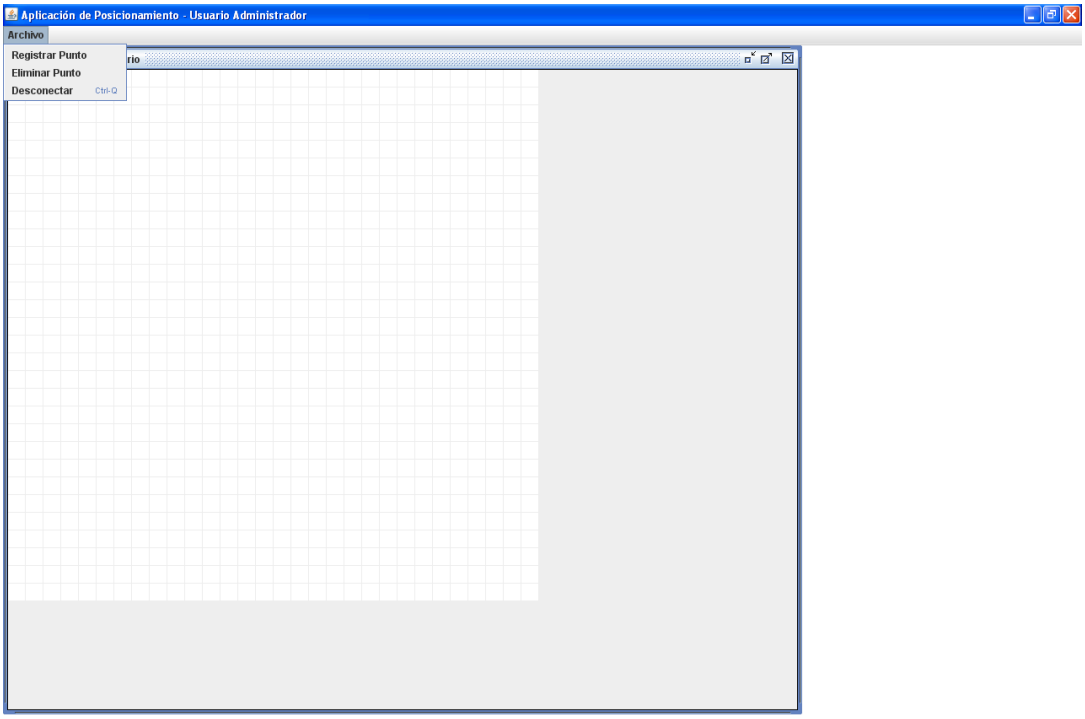


Figura 42. Ventana del panel del perfil Administrador

En primer lugar, la opción Desconectar cierra la conexión y presenta de nuevo la pantalla de inicio.

En el caso de elegir la opción de creación de puntos se presenta una ventana emergente con un formulario en el que el usuario debe introducir las coordenadas del punto a crear, así como una breve descripción si lo desea. Si se pulsa el botón aceptar, internamente el sistema obtiene la potencia media y registra el punto, representando el mismo en las casillas correspondientes de la cuadrícula que se ha citado anteriormente. Se pueden crear tantos puntos como se desee. Esta ventana se presenta en la Figura 43

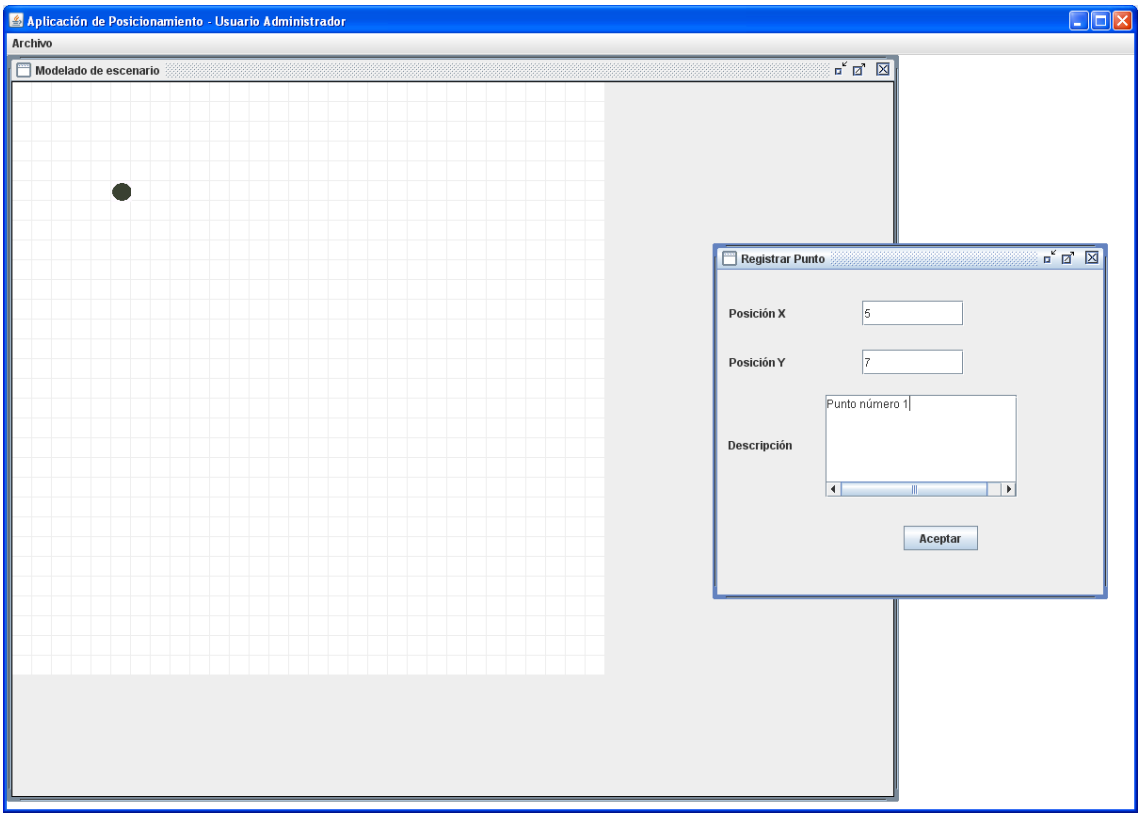


Figura 43. Ventana con la funcionalidad de registro de puntos

Por último, en el caso de elegir la opción de eliminación de puntos, se presenta también una ventana emergente como la de la Figura 44 en la que aparece un desplegable con los puntos registrados hasta el momento, por medio de sus coordenadas, de forma que, por ejemplo, si previamente se ha registrado un punto en las coordenadas (3,4), éstas aparecerán en el citado desplegable. De esta manera, el usuario puede saber de forma muy intuitiva qué punto eliminar del desplegable. Una vez elegidas las coordenadas del punto que desea borrar debe pulsar el botón Aceptar. El proceso, internamente borra los datos correspondientes y elimina el punto de la cuadrícula.

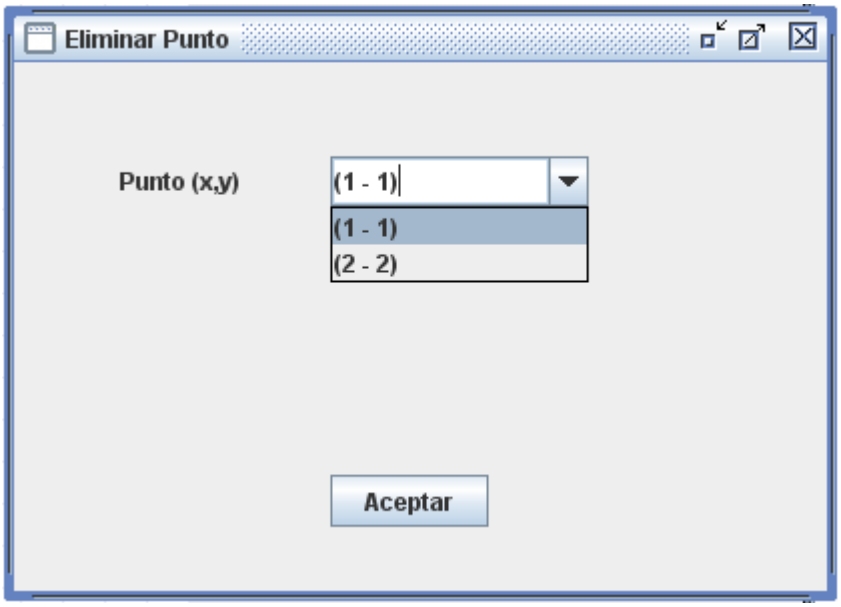


Figura 44. Ventana de eliminación de puntos

- Por otro lado, en el caso de que el usuario se haya autentificado como perfil Consultor, se presenta una nueva ventana con la opción "Archivo" en la barra de menú. Si se pulsa en este elemento se despliegan las opciones de Crear Red y Desconectar. Cabe destacar que, siguiendo el orden natural del funcionamiento de la aplicación, para crear la red previamente el usuario Administrador debe haber modelado el escenario con los puntos pertinentes.

  La opción de desconexión, como en el caso anterior, cierra la conexión y presenta de nuevo la pantalla de inicio.

  La funcionalidad de creación de red presenta en pantalla un panel como el de la Figura 45 en el que se dibuja una cuadrícula con casillas en blanco teniendo a la derecha de dicho panel pequeños bloques en los que se podrán realizar las distintas operaciones permitidas para el usuario Consultor, a saber: Crear Nodo, Borrar Nodo, Crear Enlace, Borrar Enlace.

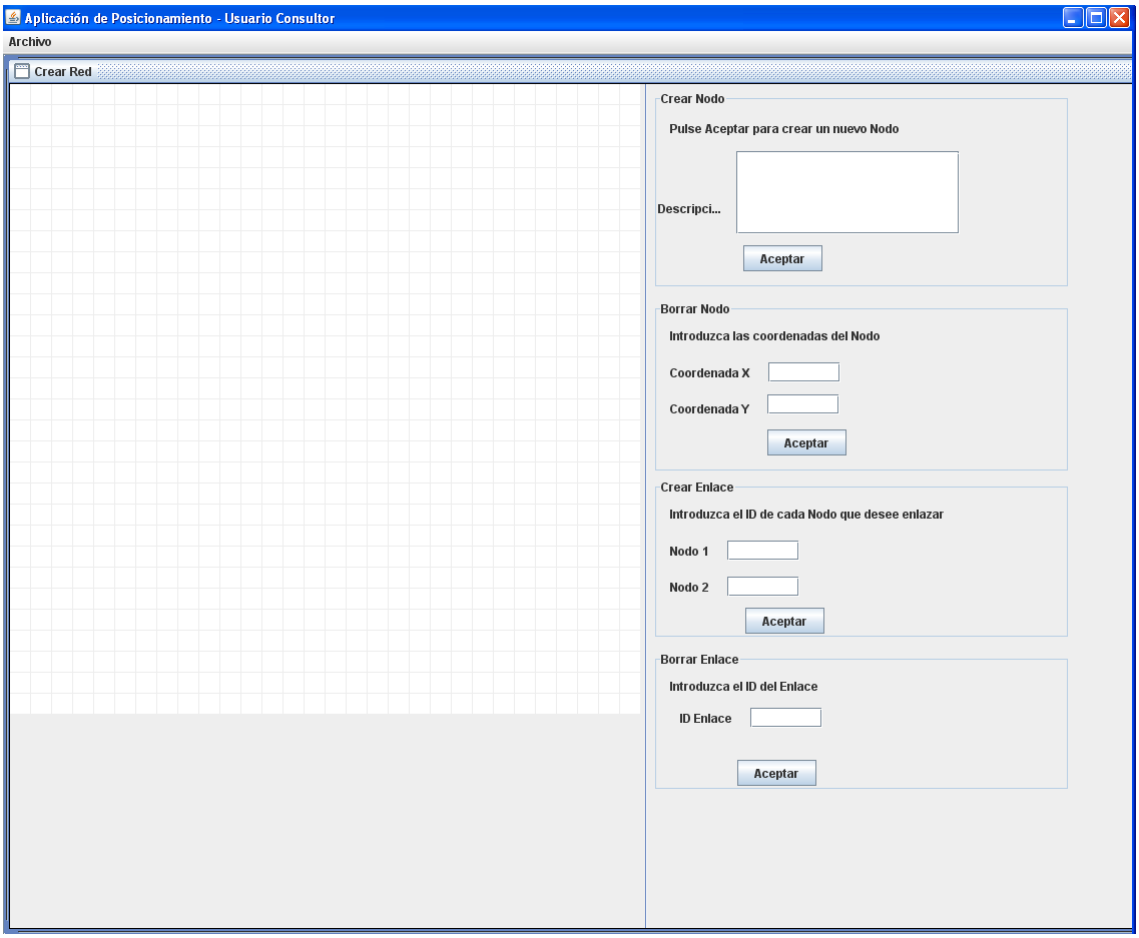


Figura 45. Ventana con el panel del perfil Consultor

Crear Nodo: para crear un nodo el usuario puede introducir una descripción antes de pulsar el botón Aceptar. Una vez pulsado este botón, el sistema internamente obtiene la potencia media del usuario en ese instante y realiza el proceso pertinente para registrar este nodo en el punto correspondiente. Inmediatamente el nodo quedará representando en las casillas de la cuadrícula en las que ha sido asignado, junto con su identificador, tal y como se puede comprobar en la Figura 46.

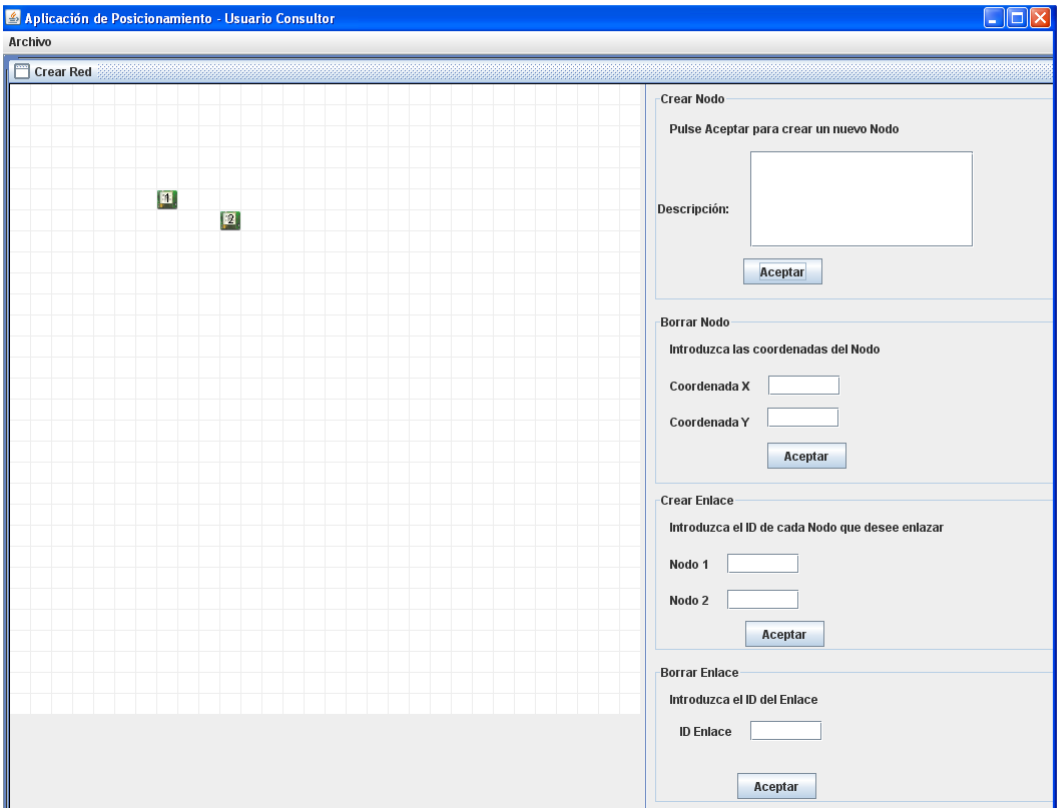


Figura 46. Ventana con dos nodos dibujados en la cuadrícula

Borrar Nodo: para borrar un nodo, el usuario debe facilitar el identificador del mismo. Éste se encuentra dibujado en el centro de la imagen del nodo en la cuadrícula. Una vez pulsado el botón Aceptar se procesa el borrado, desapareciendo por tanto el nodo de la red.

Crear Enlace: el usuario puede crear enlaces en función de los nodos existentes, es decir, para crear un enlace se debe indicar los identificadores del nodo de origen y el nodo de fin que quedarán unidos por el enlace. Una vez pulsado el botón Aceptar, queda registrado el enlace y además, en la cuadrícula, se enlazan los nodos correspondientes con una línea roja, dibujándose debajo de ésta el identificador del enlace, tal y como queda representado en la Figura 47.

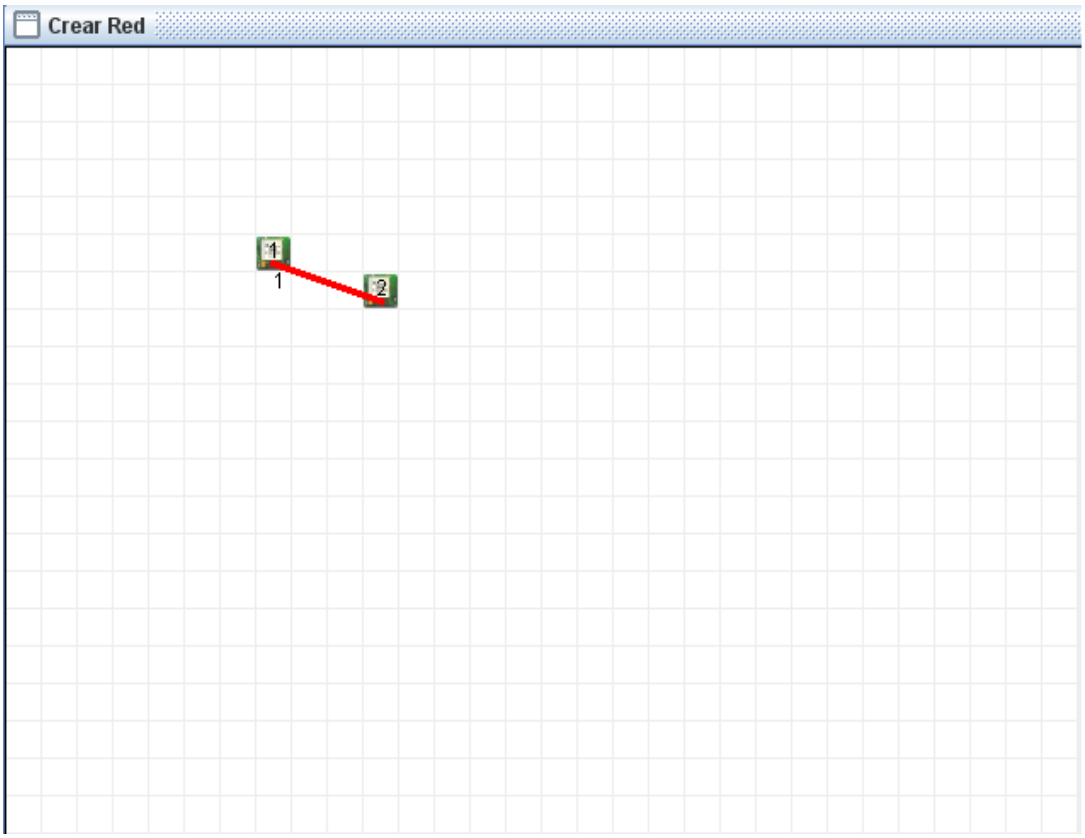


Figura 47. Cuadrícula con la representación gráfica de dos nodos y un enlace que los une

Borrar Enlace: para borrar un enlace, al igual que en el caso de borrado de nodos, el usuario debe introducir el identificador del enlace. Una vez pulsado el botón Aceptar se realiza el proceso de borrado en el que se elimina el enlace del sistema y a su vez, de la representación de la cuadrícula.